UNIVERSITÀ DI PISA

FACOLTÀ DI INGEGNERIA

RELAZIONE PER IL CONSEGUIMENTO DELLA LAUREA IN INGEGNERIA GESTIONALE

"Valutazione di linee produttive asincrone e ottimizzazione del problema di allocazione dei buffer tramite l'uso di algoritmi meta euristici"

RELATORI

Prof. Ing. Marcello Braglia
Dipartimento di Ing.Meccanica, Nucleare
e della Produzione

Dott. Ing. Marco Frosolini
Dipartimento di Ing.Meccanica, Nucleare
e della Produzione

Dott.Ing. Francesco Zammori
Dipartimento di Ing.Meccanica, Nucleare
e della Produzione

IL CANDIDATO

Anthony Ferro

Anno Accademico 2008-2009

Abstract

Valutazione di linee produttive asincrone e ottimizzazione del problema di allocazione dei buffer tramite l'uso di algoritmi meta euristici.

L'elaborato descrive le fasi di sviluppo di uno strumento operativo che permette la risoluzione del problema di allocazione dei buffer (BAP) su linee produttive asincrone.
Tale lavoro ha richiesto in primo luogo l'identificazione e l'implementazione di un opportuno tool di valutazione di linee produttive, sviluppato nel nostro caso partendo da un algoritmo di decomposizione chiamato ADDX. La fase successiva è consistita nell'identificazione di un idoneo algoritmo di ottimizzazione; essa è stata realizzata per mezzo di un'analisi di Benchmarking condotta fra due algoritmi meta euristici chiamati "Harmony Search" e "Degraded Ceiling". In ultimo luogo si è passati all'implementazione del software finale che permette di sintetizzare le funzioni svolte dagli strumenti precedentemente descritti, raggiungendo così gli obiettivi preposti.

Valuation of non-homogeneous lines and optimization of buffer allocation problem using meta heuristic algorithms.

This elaborate describes the activities carried out to develop an operative tool of evaluation of unreliable non homogeneous lines and optimization of buffer allocation problem (BAP). The building of this instrument required a preliminary phase of research where an evaluation tool was developed, modifying an already existing algorithm called ADDX, based on "decomposition" techniques. In the next step we defined an optimization tool, after an activity of Benchmarking between two meta heuristic algorithms of local search called "Degraded Ceiling" and "Harmony Search", modified to be applied to our problem. Last we developed a conclusive software that allowed us to use all the functions of the instruments described meeting the objectives set before.

0 Introduzione

Con il termine “linea produttiva” si fa riferimento ad un sistema composto da un insieme di macchine connesse in serie e separate da opportuni magazzini intermedi detti “buffers”.

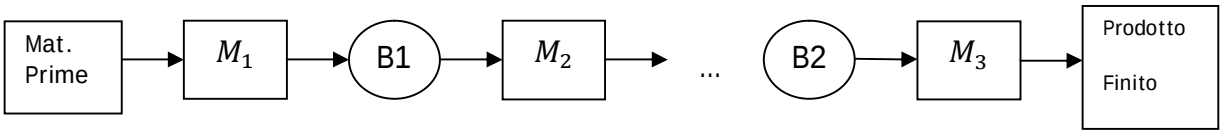


Figura 1, Linea Produttiva

Come mostrato in Figura 1, il flusso della materia prima partendo dalla macchina di inizio linea, continua passando per i successivi buffer e le successive macchine fino a confluire nel magazzino prodotto finito.

Ogni macchina è contraddistinta da una propria capacità produttiva $c = \left[\frac{PZ}{Unità\ di\ Tempo}\right]$, grandezza che esprime la capacità teorica di una macchina di generare un determinato output nel tempo.

Partendo dall’analisi delle capacità produttive delle macchine che compongono una linea è possibile generare la seguente classificazione:

- Linee produttive di tipo sincrono/bilanciate: tutte le macchine sono contraddistinte dallo stesso tempo ciclo ($C_i = C$).
- Linee produttive di tipo asincrono/non bilanciate: le macchine che le compongono sono contraddistinte da tempi ciclo differenti tra loro ($C_i \neq C_j \ \ i \neq j$)

Un classico esempio di linee sincrona è dato dalle linee transfer automatiche, in cui tutte le macchine sono caratterizzate dallo stesso tempo ciclo e i sistemi di trasferimento (vedi nastri trasportatori) cadenzano con la stessa frequenza l’affluenza alle macchine di nuovi pezzi da lavorare.

Una linea asincrona invece potrebbe essere rappresentata da una linea a trasferimento manuale, con macchine che svolgono attività delineate da differenti tempo ciclo e in cui il WIP viene accumulato fra le varie stazioni.

Nella nostra trattazione il tempo ciclo delle macchine viene sempre ipotizzato deterministico (dunque costante).

Le macchine di una linea sono inoltre descritte da altri due parametri fondamentali. Tali parametri sono il tasso di guasto $\lambda(t)\left[\frac{Guasti}{Tempo}\right]$ e il tasso di riparazione $\mu(t)\left[\frac{Riparazioni}{Tempo}\right]$; facendo ipotesi che tali quantità risultino essere costanti nel tempo si avrà che la funzione di affidabilità delle macchine considerate potrà essere approssimata ad una funzione di tipo esponenziale ($R(t) = e^{-\lambda}$).

Inoltre in questo modo potremo valutare tali parametri partendo dall' MTBF (Mean time between Failures) e dell'MTTR (Mean time to Repair), due quantità che rappresentano rispettivamente il tempo medio che occorre fra due guasti successivi su una stessa macchina e il tempo medio di riparazione degli stessi.

Avremo infatti che $\lambda = \frac{1}{MTBF}$ e $\mu = \frac{1}{MTTR}$

I dati relativi ai tassi di guasto e ai tempi medi di riparazione possono essere rilevati direttamente sul campo attraverso opportune tecniche utilizzate in ambito manutentivo.

Da quanto detto si capisce come le prestazioni di una macchina dipendano fortemente dalla sua capacità intrinseca di generare output, ma anche dal suo rendimento.

Questo fa si che in linee composte da macchine soggette a guasti non sia possibile utilizzare l'accezione classica di "Collo di Bottiglia" in quanto l'effettiva macchina più critica del sistema potrebbe essere caratterizzata da elevati valori di c sfruttati minimamente a causa di rendimenti poco elevati.

Per ovviare al problema dei fermi macchina, usualmente si interpongono fra le varie stazioni di una linea dei piccoli magazzini intermedi: l'effetto dei buffer è quello di far si che il fermo di una o più macchine non generi la fermata del sistema.

Tutti i risultati a cui si perverrà saranno riferiti ad una condizione di funzionamento a regime della linea: solitamente tale condizione viene soddisfatta quando tutti i buffer intermedi sono pieni almeno per metà.

La fase di warm-up verrà rappresentata dal periodo necessario ad arrivare a tale condizione.

Il dimensionamento corretto dei buffer è uno dei problemi fondamentali nell'ambito dell'ottimizzazione delle linee produttive, sia bilanciate che non bilanciate.

E' possibile intervenire agendo sui buffer in due differenti modi: o incrementandone notevolmente le dimensioni o sfruttando gli spazi disponibili dimensionandoli opportunamente.
Contando che lo spazio disponibile all'interno degli stabilimenti produttivi è una risorsa scarsa possiamo affermare con certezza che aumentare notevolmente le dimensioni dei buffer sia una soluzione praticamente impossibile da realizzare.
Al contrario, l'allocazione corretta dello spazio disponibile è una soluzione pratica che può portare anche a notevoli incrementi della produttività di sistema (fino al 10%). Il problema legato a tale metodologia di ottimizzazione è che non esistono metodi pratici con cui attuarla e spesso i buffer vengono dimensionati basandosi sull'esperienza o tramite metodologie basate sul "Trial and Error".
Fino ad oggi sono stati sviluppati due modelli base di rappresentazione delle linee produttive, basati sulla teoria delle catene di Markov e denominati Modello Discreto e Modello Continuo.
Il modello discreto, sviluppato da Buzacott, schematizza la linea attraverso un processo di Markov a stati discreti, ma ha trovato opportune applicazioni solamente nel caso delle linee di tipo "balanced", composte dunque da macchine contraddistinte tutte dallo stesso tempo ciclo.
Il modello continuo, proposto da Zimmem e Sevast'yanov, attraverso l'approssimazione del flusso discreto delle parti lungo linea ad un flusso di tipo continuo, ha permesso lo studio di linee più complesse, soprattutto quelle di tipo non bilanciato e ha aperto la strada alla formalizzazione di modelli con cui valutare le prestazioni di linee composte da un numero elevato di macchine.
In ultimo luogo si ricorda che l'ottimizzazione del buffer all'interno di una linea produttiva è un'attività che viene realizzata attraverso l'utilizzo simultaneo di due differenti tool :

- Strumenti di valutazione della linea, che permettono una volta definite le sue caratteristiche e il valore di un possibile buffer, di ricavarne le prestazioni fondamentali.
- Strumenti di ottimizzazione, che permettono di facilitare la ricerca della soluzione ottima del problema all'interno dell'intero spazio di ricerca.

1 Valutazione Performance di Linee Base

Le macchine operative sono solitamente valutate per mezzo di un parametro prestazionale definito produttività o Throughput che per definizione è dato da: $TP = \eta * c$, con $\eta = \left(\frac{\mu}{\lambda+\mu}\right)$.

Tale parametro misura la capacità effettiva di una macchina di fornire un determinato output e viene valutato come prodotto fra l'efficienza della macchina stessa e la sua capacità produttiva di targa. Il TP altro non è che una lettura corretta della capacità produttiva, rispetto all'efficienza della macchina stessa. Più elevato sarà il tempo di servizio medio e più elevato sarà lo sfruttamento delle possibilità della macchina, più ridotto sarà il tempo medio di riparazione e minori saranno gli effetti di un guasto sulla sua capacità di realizzare un output.

Il TP è un parametro che ha una valenza del tutto generale e che dunque può essere usato tranquillamente anche a livello di sistema.

Nel momento in cui si passa dall'idea di singola macchina a quella di linea produttiva, la sua valutazione diviene assai più articolata perché la linea è un sistema complesso in cui oltre al comportamento delle singole macchine che la compongono dovremo tenere conto di altri fattori più generali.

In primo luogo si avrà che l'asincronia fra le varie macchine comporterà un disallineamento fra le produzioni delle stesse che si rifletteranno in seguito in una discontinuità della produzione.

In secondo luogo, su una linea potranno manifestarsi guasti singoli o molteplici che frazioneranno il flusso in output e ridurranno nettamente la produttività globale.

Questo comporta che l'analisi delle prestazioni di una linea produttiva richiede lo sviluppo di opportuni modelli che ne riassumano le caratteristiche fondamentali e che permettano effettivamente di riprodurne il comportamento.

Fino ad oggi sono stati implementati molti e differenti sistemi per riuscire a raggiungere tale obiettivo, che si rifanno essenzialmente a due macrotecniche di descrizione:

- Modelli basati su simulazione
- Modelli di tipo analitico

Simulazione

La simulazione sta acquisendo sempre più importanza nell'analisi dei processi produttivi. Tali sistemi sono infatti caratterizzati da elevata complessità, numerose inter-relazioni tra i diversi processi che li attraversano, guasti dei segmenti, indisponibilità, stocasticità dei parametri del sistema.

Nei modelli da noi considerati ad esempio i parametri stocastici sono rappresentati dai tempi di guasti e di riparazione e teoricamente anche dai tempi ciclo delle macchine.

Questo fa si che il ritmo produttivo di una linea non sia costante, ma funzione dell'andamento di tali parametri.

Un ipotetico progettista di impianti industriali e i responsabili delle operations possono certamente avere interesse a valutare con anticipo l'effetto delle loro scelte su tali sistemi complessi, in termini, ad esempio, di capacità di produzione, tempo di attraversamento, scorte, blocchi.

La simulazione, consentendo l'analisi della realtà ad un elevato livello di dettaglio e padroneggiando facilmente la complessità del sistema, fa si che alla fine sia possibile ottenere un gran numero di informazioni utili.

Il prezzo da pagare per tale completezza è ovviamente il tempo, infatti le operazioni di programmazione (dunque tutte le fasi a monte della vera e propria analisi simulativa) sono assai lunghe, affinché si possano ottenere dei dati sufficientemente sensati e tali da dare la possibilità di ottenere un modello della realtà ad essa aderente.

Al fine di poter procedere correttamente per avere un modello di simulazione utile e funzionante è opportuno procedere con una serie di passi:

Definizione degli obiettivi e delle problematiche da esaminare: un' attenta analisi del problema consente di circoscriverne l'esame riducendo il successivo tempo di analisi,ma soprattutto un'analisi approfondita permette di comprendere al meglio il problema da risolvere, agevolando le fasi successive di creazione dello stesso.

Stesura di un modello concettuale: consiste nella comprensione e modellazione del sistema produttivo che si intende simulare; questa fase è particolarmente importante in quanto definirà il comportamento dei diversi flussi di materiale e di informazioni che attraverseranno il modello.

Validazione del modello concettuale: si tratta di un confronto con la direzione dell'impresa e con gli operatori per assicurarsi della capacità del modello di offrire un'immagine consistente della realtà.

***Analisi dei dati in ingresso*:** la raccolta e l'analisi dei dati che diverranno la base per la definizione dei parametri di funzionamento del sistema (ad esempio: i diversi tempi ciclo di una singola macchina). Le analisi sono quasi sempre di tipo probabilistico e servono per leggere in maniera critica i dati di partenza.

***Scrittura del modello in termini matematici* :** è la fase in cui si implementa il modello creato su apposite piattaforme software che permetteranno al modello stesso di girare successivamente,

Definizione di un piano degli esperimenti(DOE): una singola iterazione ("run") di simulazione non ha alcun significato; rappresenta solo una delle possibili evoluzioni del sistema. È quindi opportuno effettuare diversi "run" per poi analizzare i parametri in uscita. La lunghezza della singola iterazione e il numero delle iterazioni vengono determinate in questa fase.

***Analisi dei dati in uscita*:** dopo aver raccolto i dati relativi ai parametri, depurati da eventuali transitori è possibile creare degli intervalli di confidenza ovvero stimare il "range" di valori in cui i parametri che analizzano il problema proposto al primo passaggio possono oscillare.

Elementi caratteristici di un modello di simulazione

Entità : Le entità sono gli elementi "trattati" dal processo; tali "oggetti" hanno la caratteristica di essere "temporanei", e di subire passivamente le trasformazioni. Ad esempio, in un'impresa di lavorazioni meccaniche, i semilavorati e le materie prime, che devono essere fresati, spianati etc possono essere modellizzati come "entità".

Operazione: rappresenta una delle trasformazioni che avranno luogo sull'entità. Possono essere individuati due cicli di operazioni:

Il ciclo macchina: attinente agli stati (vedi) ed operazioni che la macchina attraverserà, ovvero l'insieme di tutte le possibili successioni di operazioni e attese.

Il ciclo pezzo: rappresenta il percorso delle entità nel sistema, le macchine visitate e le operazioni subite

Macchine: rappresentano gli elementi "fissi" del sistema, la cui definizione degli stati definisce univocamente la situazione generale del sistema, e delle quali sono di rilevanza per l'analista soprattutto le prestazioni. Le macchine possono essere fisiche, ed in questo caso ci si riferisce a macchine realmente presenti nel sistema da modellare, o "logiche", ed in questo caso compiono operazioni "fittizie" fisicamente, ma presenti logicamente nel sistema (ad esempio, il controllo di quantità in ingresso nell'impianto non ne provoca trasformazioni "fisiche" ma lo "trasforma" da "lotto da controllare" a "lotto controllato").
Stati: gli stati sono delle variabili (di tipo vario: possono essere numeri o valori logici) che descrivono lo stato del sistema e delle sue componenti, per ogni istante di tempo.
Eventi: fenomeni che modificano lo stato del sistema (ad esempio, la fine di una lavorazione modifica lo stato di una macchina da "occupata" a "libera").
Code: insiemi di entità che non possono accedere alle trasformazioni successive in quanto la macchina risulta occupata.
Attributi: proprietà permanenti di un insieme di entità o di una macchina.
Orologio locale: orologio che contiene, a livello di singola macchina, l'istante di tempo che identifica la fine della lavorazione in corso.
Orologio generale: orologio che regola lo scorrere generale del tempo di simulazione
Nella nostra trattazione, tutte le simulazioni sono state condotte per mezzo di un software applicativo di tipo general pourpose chiamato SIMUL8: tale software permette di eliminare o semplificare le prime fasi della simulazione, proponendo modelli pre-realizzati e macchine già settate. Questo ha permesso di risparmiare un elevata quantità di tempo, ottenendo risultati molto efficaci.

SISTEMI ANALITICI

I sistemi di tipo analitico permettono di valutare le prestazioni di una linea asincrona sviluppando opportuni modelli matematici, spesso complessi quanto quelli simulativi, ma che una volta realizzati permettono di ottenere risultati in tempi globali assai più ridotti.

Inoltre tali sistemi presentano un ulteriore vantaggio fondamentale: essi permettono, una volta affiancati da opportune strumenti di ricerca, di essere utilizzati per analisi di ottimizzazione, mentre la simulazione non prevede ulteriori utilizzi se non quelli di semplice calcolo dei parametri prestazionali.

In parole povere il sistema di valutazione può divenire mezzo strumentale per una successiva analisi migliorativa di una linea considerata.

La produttività di una linea è dunque funzione di:

- Numero delle macchine che compongono la linea produttiva.
- Parametri caratteristici della linea (λ, μ, c).
- Capacità dei buffer compresi fra le macchine che la compongono.

L'obiettivo che ci prefiggiamo di raggiungere è duplice, in primis quello di stabilire un sistema che presi in input i parametri caratteristici di una linea ci permetta di definirne il Throughput globale e in secondo luogo quello di realizzare un sistema analitico che data una linea produttiva permetta di definirne l'allocazione ottimale degli spazi fra i vari buffer in modo tale da incrementarne le prestazioni globali.

1.1 Modello di Gershwin per Linee Asincrone di Base

Una linea di base è un sistema produttivo composto da due sole macchine, separate da un opportuno buffer: esse rappresentano il caso più elementare di studio di sistema produttivo, facilmente modellizzabile per mezzo di opportune catene di Markov.

Il Throughput di una linea asincrona di base può essere calcolato utilizzando il modello di Gershwin che si basa sulle seguenti ipotesi fondamentali:

(i) La distribuzione dei tempi di guasto sia esponenziale con tasso di guasto λ,

(ii) La distribuzione dei tempi di riparazione sia esponenziale con tasso di riparazione μ.

(iii) Il tempo ciclo delle macchine sia deterministico.

(iv) Il buffer fra le macchine sia finito e di valore N.

In questo caso è infatti possibile modellare il sistema come un processo di Markov a tempo continuo in cui ogni possibile stato è rappresentabile dalla terna:

$$\mathbf{S}(t) = \{n(t), s_1(t), s_2(t)\}$$

dove $n(t)$, $s_1(t)$, $s_2(t)$ rappresentano rispettivamente la dimensione della coda nel buffer intermedio e lo stato della prima e della seconda macchina del sistema.

Si noti che $0 \leq n(t) \leq N$ e che $s_i(t) \in \{0,1\}$ dato che una macchina è rotta o è funzionante.

Procedendo in tal modo è possibile risolvere in forma chiusa il sistema di equazioni differenziali che regola il sistema sopra descritto ed ottenere il valore di tutte le probabilità di stato del sistema $\mathbf{P}(t) = \mathbf{P}_{ijk}(t) = \mathbf{P}(n = i, s_1 = j, s_2 = k)$. Queste ultime permettono infine di valutare tutti i principali parametri.

Ne consegue dunque il modello sotto riportato:

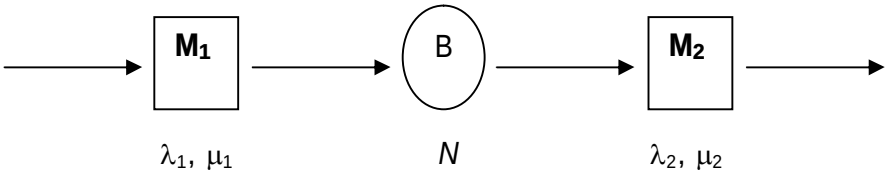


Figura 2, Linea Base

Nel caso in cui si stia studiando una linea di tipo asincrono i tempi cicli τ_1 e τ_2 delle due macchine (e quindi le rispettive capacità produttive $c_1 = 1/\tau_1$; $c_2 = 1/\tau_2$) pur restando determininstici differiscono fra loro.

In questo caso anche se l'efficienza di macchina singola ($\epsilon_i = \mu_i/(\mu_i + \lambda_i)$) non cambia, come valore di performance è necessario parlare di throughput TR_i [pezzi/intervallo ditempo] definito nel modo seguente:

$$TR_i = c_i\left(\frac{\mu_i}{\mu_i + \lambda_i}\right)$$

Utilizzando le catene di Markov è possibile determinare esattamente tutti i parametri di interesse.

Probabilità di stato a regime

Definiamo innanzitutto le seguenti costanti:

$$K_1 = \frac{(E_1+E_4)}{2} + \frac{\sqrt{(E_1-E_4)^2 + 4E_2E_3}}{2}$$

$$K_2 = \frac{(E_1+E_4)}{2} - \frac{\sqrt{(E_1-E_4)^2 + 4E_2E_3}}{2}$$

$$B_1 = C_0\left(F_4 - \frac{F_2F_5}{2M_2}\right)$$

$$B_2 = \begin{cases} C_0\left(\frac{F_1F_3F_5}{2E_2} - F_1F_4\right)e^{(K_1-K_2)N}, & se \;\; c_1 < c_2 \\ C_0\left(\frac{F_1F_3F_5}{2E_2} - F_1F_4\right), & se \;\; c_1 < c_2 \end{cases}$$

$$B_3 = F_6C_0$$

$$B_4 = F_7C_0$$

$$E_1 = \frac{\lambda_2(c_1\mu_1+c_2\mu_2)-\mu_2(c_2-c_1)(\lambda_1+\lambda_2+\mu_1+\mu_2)}{c_1(c_2-c_1)(\mu_1+\mu_2)}$$

$$E_2 = \frac{-\lambda_2(c_1\mu_1+c_2\mu_2)}{c_1(c_2-c_1)(\mu_1+\mu_2)}$$

$$E_3 = \frac{-\lambda_1(c_1\mu_1+c_2\mu_2)}{c_2(c_2-c_1)(\mu_1+\mu_2)}$$

$$E_4 = \frac{\lambda_1(c_1\mu_1+c_2\mu_2)+\mu_1(c_2-c_1)(\lambda_1+\lambda_2+\mu_1+\mu_2)}{c_2(c_2-c_1)(\mu_1+\mu_2)}$$

$$C_0 = \frac{1}{F_4F_8 + F_5F_9 + F_6 + F_7}$$

$$F_1 = \begin{cases} \dfrac{c_2(\mu_1+\mu_2+\lambda_1)(E_1-E_4-K_1+K_2)+2\lambda_1 c_1 E_2}{c_2(\mu_1+\mu_2+\lambda_1)(E_1-E_4+K_1-K_2)+2\lambda_1 c_1 E_2}, & se\ c_1 < c_2 \\ \dfrac{c_2\lambda_2(E_1-E_4-K_1+K_2)+2(\mu_1+\mu_2+\lambda_2)c_1 E_2}{c_2\lambda_2(E_1-E_4+K_1-K_2)+2(\mu_1+\mu_2+\lambda_2)c_1 E_2}, & se\ c_1 > c_2 \end{cases}$$

$$F_2 = E_1 - E_4 - (K_1 - K_2)$$

$$F_3 = E_1 - E_4 + (K_1 - K_2)$$

$$F_4 = \frac{\lambda_1}{\mu_1+\mu_2} + \frac{c_2}{c_2-c_1}$$

$$F_5 = \frac{\lambda_2}{\mu_1+\mu_2} - \frac{c_1}{c_2-c_1}$$

$$F_6 = \begin{cases} \dfrac{2E_2c_1(\mu_1+\lambda_1)(\mu_1+\mu_2+\lambda_2)\left(1-F_1e^{(K_1-K_2)N}\right)+c_2\lambda_2(\lambda_2+\mu_2)\left(-F_2+F_1F_3e^{(K_1-K_2)N}\right)}{2E_2\lambda_2\mu_1(\lambda_1+\lambda_2+\mu_1+\mu_2)}, & se\ c_1 < c_2 \\ \dfrac{c_2(F_1F_3-F_2)}{(2\mu_1E_2)}, & se\ c_1 > c_2 \end{cases}$$

$$F_7 = \begin{cases} \dfrac{c_1(1-F_1)e^{K_1N}}{\mu_2}, & se\ c_1 < c_2 \\ \dfrac{e^{K_1N}[c_2(\mu_2+\lambda_2)(\lambda_1+\mu_1+\mu_2)\left(-F_2+F_1F_3e^{(K_2-K_1)N}\right)+2E_2c_1\lambda_1(\lambda_1+\mu_1)(1-F_1e^{(K_2-K_1)N})]}{2E_2\lambda_1\mu_2(\lambda_1+\lambda_2+\mu_1+\mu_2)}, & se\ c_1 > c_2 \end{cases}$$

$$F_8 = \begin{cases} \dfrac{e^{K_1N}-1}{K_1} - \dfrac{e^{K_2N}-1}{K_2}F_1e^{(K_1-K_2)N}, & se\ c_1 > c_2 \\ \dfrac{e^{K_1N}-1}{K_1} - \dfrac{e^{K_2N}-1}{K_2}F_1, & se\ c_1 < c_2 \end{cases}$$

$$F_9 = \begin{cases} \dfrac{1-e^{K_1N}}{K_1}\cdot\dfrac{F_2}{2E_2} + \dfrac{e^{K_2N}-1}{K_2}\cdot\dfrac{F_1F_3e^{(K_2-K_1)N}}{2E_2}, & se\ c_1 > c_2 \\ \dfrac{1-e^{K_1N}}{K_1}\cdot\dfrac{F_2}{2E_2} + \dfrac{e^{K_2N}-1}{K_2}\cdot\dfrac{F_1F_3}{2E_2}, & se\ c_1 < c_2 \end{cases}$$

Utilizzando tali valori la distribuzione di probabilità per lo stato $h \in \{0,1,\ldots,N\}$ vale:

$$f_{H,I,J}(h,1,0) = C_0\left[e^{K_1h} - F_1e^{(K_1-K_2)N}\cdot e^{K_2h}\right]$$

$$f_{H,I,J}(h,0,1) = C_0\left[-\frac{F_2}{2E_2}e^{K_1h} + \frac{F_1F_3}{2E_2}e^{(K_1-K_2)N}\cdot e^{K_2h}\right]$$

$$f_{H,I,J}(h,1,1) = \frac{c_1}{c_2-c_1}f_{H,I,J}(h,1,0) - \frac{c_2}{c_2-c_1}f_{H,I,J}(h,0,1)$$

$$f_{H,I,J}(h,0,0) = \frac{\lambda_1}{\mu_1+\mu_2}f_{H,I,J}(h,1,0) + \frac{\lambda_2}{\mu_1+\mu_2}f_{H,I,J}(h,0,1)$$

Attraverso i valori così determinati è possibile valutare le probabilità di stato a regime:

Se $c_1 < c_2$,

$$P_{0,1,1} = \frac{c_1(\lambda_1 + \mu_1 + \mu_2) f_{H,I,J}(0,1,0) - c_2\lambda_2 f_{H,I,J}(0,0,1)}{\lambda_2(\lambda_1 + \lambda_2 + \mu_1 + \mu_2)} \qquad (\mathbf{1.17a})$$

$$P_{0,1,0} = 0 \qquad (1.18a)$$

$$P_{0,0,1} = \frac{(\mu_1 + \mu_2)[c_1\lambda_1 f_{H,I,J}(0,1,0) + c_2\lambda_2 f_{H,I,J}(0,0,1)]}{\lambda_2\mu_1(\lambda_1 + \lambda_2 + \mu_1 + \mu_2)} \qquad (\mathbf{1.19a})$$

$$P_{0,0,0} = \frac{c_1\lambda_1 f_{H,I,J}(0,1,0) + c_2\lambda_2 f_{H,I,J}(0,0,1)}{\mu_1(\lambda_1 + \lambda_2 + \mu_1 + \mu_2)} \qquad (1.20a)$$

$$P_{N,1,1} = 0 \qquad (1.21a)$$

$$P_{N,1,0} = \frac{c_1 f_{H,I,J}(N,1,0) - c_2 f_{H,I,J}(N,0,1)}{\mu_2} \qquad (\mathbf{1.22a})$$

$$P_{N,0,1} = 0 \ (1.23) \qquad (1.23a)$$

$$P_{N,0,0} = \frac{\lambda_1[c_1 f_{H,I,J}(N,1,0) - c_2 f_{H,I,J}(N,0,1)]}{\mu_2(\mu_1 + \mu_2)} \qquad (1.24a)$$

Se $c_1 > c_2$,

$$P_{0,1,1} = 0 \qquad (1.17b)$$

$$P_{0,1,0} = 0 \qquad (1.18b)$$

$$P_{0,0,1} = \frac{-c_1 f_{H,I,J}(0,1,0) + c_2\lambda_2 f_{H,I,J}(0,0,1)}{\mu_1} \qquad (1.19b)$$

$$P_{0,0,0} = \frac{c_1}{\mu_1} f_{H,I,J}(0,1,0) \qquad (1.20b)$$

$$P_{N,1,1} = \frac{c_2(\lambda_1 + \mu_1 + \mu_2) f_{H,I,J}(N,0,1) - c_1\lambda_1 f_{H,I,J}(N,1,0)]}{\lambda_1(\lambda_1 + \lambda_2 + \mu_1 + \mu_2)} \qquad (1.21b)$$

$$P_{N,1,0} = \frac{(\mu_1 + \mu_2)[c_1\lambda_1 f_{H,I,J}(N,1,0) + c_2\lambda_2 f_{H,I,J}(N,0,1)]}{\lambda_1\mu_2(\lambda_1 + \lambda_2 + \mu_1 + \mu_2)} \qquad (1.22b)$$

$$P_{N,0,1} = 0 \qquad (1.23b)$$

$$P_{N,0,0} = \frac{c_1\lambda_1 f_{H,I,J}(N,1,0) + c_2\lambda_2 f_{H,I,J}(N,0,1)]}{\mu_2(\lambda_1 + \lambda_2 + \mu_1 + \mu_2)} \qquad (1.24b)$$

La distribuzione di probabilità marginale (a regime) del valore del buffer è data da:

$$f_H(h) = B_1 e^{K_1 h} + B_2 e^{K_2 h} + B_3\delta(h) + B_4\delta(h - N), \quad h \in \{0,1,\dots,N\} \qquad (1.25)$$

Indicatori di performance

A questo punto è possibile valutare tutta una serie di indicatori di performance.

Throughput

$$= \begin{cases} TP = \dfrac{G_4 + G_5 e^{K_1 N} + G_6 e^{K_2 N}}{G_1 + G_2 e^{K_1 N} + G_3 e^{K_2 N}}, & se\ c_1 > c_2 \\ TP = \dfrac{G_4 + G_5 e^{K_1 N} + G_6 e^{K_2 N}}{G_1 + G_2 e^{K_1 N} + G_3 e^{K_2 N}}, & se\ c_1 < c_2 \end{cases} \quad (1.26)$$

dove:

$$G_1 = \begin{cases} G_1 = \mu_1 G_7^2 + \mu_1 G_7 [c_1(\mu_1 + \mu_2 + \lambda_2) - c_2(\mu_1 + \mu_2 + \lambda_1), & se\ c_1 > c_2 \\ G_1 = \mu_1 G_7^2 + \mu_1 G_7 [c_1(\mu_1 + \mu_2 + \lambda_2) - c_2(\mu_1 + \mu_2 + \lambda_1), & se\ c_1 < c_2 \end{cases} \quad (1.27)$$

$$G_2 = \begin{cases} \mu_2 \lambda_1 c_2 [(c_1 - c_2)(\mu_1 - \mu_2) - (c_2\lambda_1 + c_1\lambda_2) - G_7], & se\ c_1 < c_2 \\ \mu_1 \lambda_2 c_1 [(c_1 - c_2)(\mu_1 - \mu_2) - (c_2\lambda_1 + c_1\lambda_2) + G_7], & se\ c_1 > c_2 \end{cases}$$

$$G_3 = \begin{cases} \dfrac{\epsilon_2(c_2 - c_1\epsilon_1)G_1 + c_1\epsilon_1(1-\epsilon_2)G_2}{c_1\epsilon_1(\epsilon_2 - 1)}, se\ c_1 < c_2 \\ \dfrac{\epsilon_1(c_1 - c_2\epsilon_2)G_1 + c_2\epsilon_2(1-\epsilon_1)G_2}{c_2\epsilon_2(\epsilon_1 - 1)}, se\ c_1 > c_2 \end{cases}$$

$$G_4 = \begin{cases} c_2\epsilon_2 G_1, & se\ c_1 < c_2 \\ c_1\epsilon_1 G_1, & se\ c_1 > c_2 \end{cases}$$

$$G_5 = \begin{cases} c_1\epsilon_1 G_2, & se\ c_1 < c_2 \\ c_2\epsilon_2 G_2, & se\ c_1 > c_2 \end{cases}$$

$$G_6 = \begin{cases} c_1\epsilon_1 G_3, & se\ c_1 < c_2 \\ c_2\epsilon_2 G_3, & se\ c_1 > c_2 \end{cases}$$

$$G_7 = \sqrt{[c_1(\mu_1 + \mu_2 + \lambda_2) - c_2(\mu_1 + \mu_2 + \lambda_1)]^2 + 4c_1c_2\lambda_1\lambda_2}$$

Work In Process

$$WIP = \frac{F_4F_{10} + F_5F_{11} + F_7N}{F_4F_8 + F_5F_9 + F_6 + F_7} \qquad (1.27)$$

dove:

$$F_{10} = \begin{cases} \dfrac{1 + (K_1N - 1)e^{K_1N}}{K_1^2} + \dfrac{1 + (K_2N - 1)e^{K_2N}}{K_2^2} \cdot F_1 e^{(K_1 - K_2)N}, & se\ c_1 < c_2 \\ \dfrac{1 + (K_1N - 1)e^{K_1N}}{K_1^2} - \dfrac{1 + (K_2N - 1)e^{K_2N}}{K_2^2} \cdot F_1, & se\ c_1 < c_2 \end{cases}$$

$$F_{11} = \begin{cases} \dfrac{F_2[1 + (K_1N - 1)e^{K_1N}]}{2E_2K_1^2} + \dfrac{[1 + (K_2N - 1)e^{K_2N}]F_1F_3e^{(K_1 - K_2)N}}{2E_2K_2^2}, & se\ c_1 < c_2 \\ -\dfrac{[1 + (K_1N - 1)e^{K_1N}]F_2}{2E_2K_1^2} + \dfrac{[1 + (K_2N - 1)e^{K_2N}]F_1F_3}{2E_2K_2^2}, & se\ c_1 < c_2 \end{cases}$$

Blockage (BL_1) and Starvation (ST_2)

$$BL_1 = \epsilon_1 - \frac{TP}{c_1} = \frac{\epsilon_1 c_1 - TP}{c_1} \qquad (1.28)$$

$$ST_2 = \epsilon_2 - \frac{TP}{c_2} = \frac{\epsilon_2 c_2 - TP}{c_2} \qquad (1.29)$$

1.2 Validazione Modello di Gershwin

Il modello di Gershwin rappresenta il cardine fondamentale per le tecniche di valutazione delle prestazioni di linee complesse in quanto tutti i sistemi fino ad ora implementati fanno si che per mezzo di opportune sostituzioni e trasformazioni una linea di dimensioni elevate possa essere ricondotta ad una elementare e dunque analizzabile per mezzo di tale modello.

In letteratura esistono altri modelli alternativi: per questo motivo, prima di procedere con il suo utilizzo è stato ritenuto necessario validarlo per comprendere se continuare con il suo utilizzo o valutare possibili approcci alternativi a tale problema.

Per farlo è stato sviluppato un semplice software in linguaggio Delphi con cui sono state valutate delle linee base, su singoli valori di della capacità del buffer in esse contenuto. I valori ottenuti tramite la sperimentazione sono stati in seguito confrontati con valori calcolati per mezzo della simulazione.

Il confronto con tale sistema di valutazione non è casuale, in quanto come è stato scritto precedentemente, esso rappresenta ad oggi il sistema più preciso e dunque il miglior punto di riferimento.

Linea 1	M1	M2	Linea 2	M1	M2	Linea 3	M1	M2	Linea 4	M1	M2
λ	0,1	0,1	λ	0,1	0,1	λ	0,01	0,01	λ	0,01	0,1
μ	0,5	0,5	μ	0,5	0,5	μ	0,05	0,05	μ	0,5	0,5
c	1	0,9	c	0,9	1	c	1	0,9	c	0,9	1

L'errore medio valutato si attesta allo 0,6 %, percentuale che conferma l'elevata efficacia del modello che potrà così essere utilizzato in seguito come pilastro del sistema di valutazione di linee asincrone complesse.
Il software implementato è stato in seguito utilizzato per portare a termine un altro tipo di analisi, consistente nella valutazione dei casi precedentemente testati non più su un singolo valore delle capacità del buffer, bensì su un dominio più elevato. Tutto ciò ci ha permesso in primo luogo di tracciare l'andamento del TP di tali sistemi al variare di tale parametro e in secondo luogo di comprendere al meglio alcuni attributi intrinseci di tali sistemi.

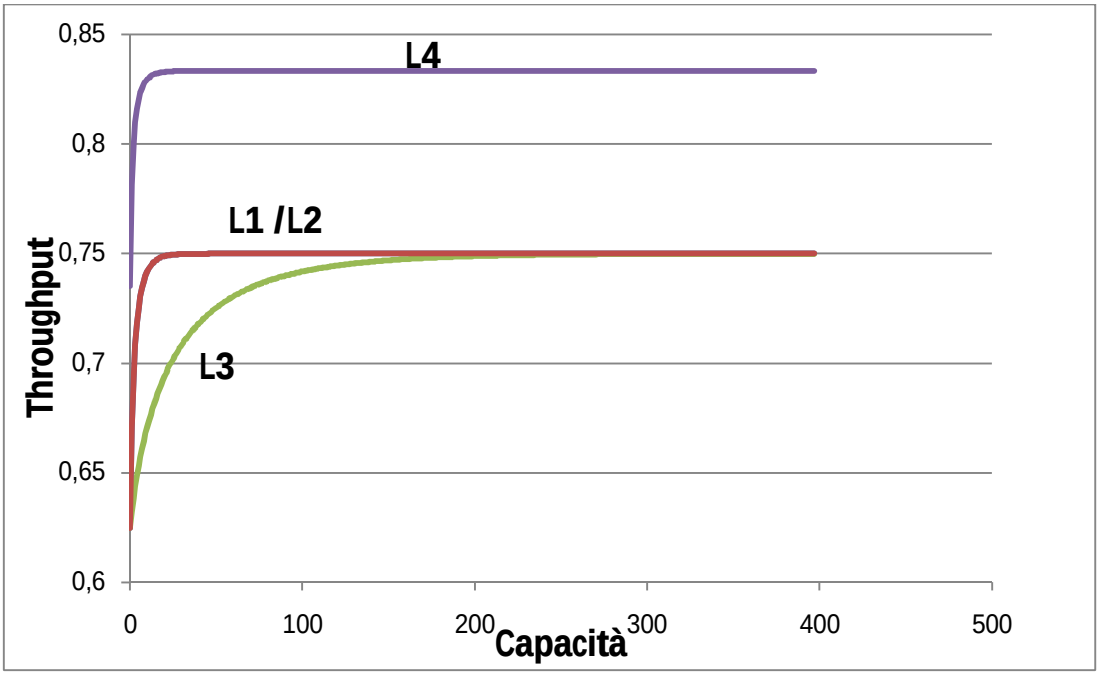


Figura 3, Rendimento Linee Base

In particolare si possono metter in evidenza le seguenti caratteristi:

- **Reversibilità:** data una linea del tipo $M2$, invertendo l'ordine delle macchine che la compongono il suo Throughput globale non cambia (casi L1 e L2).
- **Monotonicità:** il TP di una linea cresce monotonicamente. Il suo andamento non è però strettamente monotonico infatti a fronte di una crescita molto elevata iniziale il TP tende a crescere sempre meno velocemente fino a coincidere con un asintoto orizzontale sempre presente nei casi studiati.
- **Efficienza:** a parità di efficienza delle macchine che le compongono, più i tempi di fermo delle stesse sono elevati e minore è il TR della linea. Tempi di Down Time elevati possono fare decadere le prestazioni di una linea anche del 25%.

Come si può evincere dal grafico a parità di capacità produttiva una stessa linea avrà produttività crescente al crescere dei tassi di riparazione delle macchine

2 Valutazione Performance di Linee Complesse

2.1 Descrizione Linee Complesse

Una linea asincrona composta da un numero di macchine superiore a due viene definita complessa e teoricamente, può essere studiata come una sistema di base. Questo vuol dire essenzialmente che tali linee possono essere ridotte a catene di Markov equivalenti, di cui è possibile valutare le probabilità di stato a regime con cui calcolarene in seguito i parametri fondamentali, fra cui il Throughput globale e il WIP medio.

Il problema fondamentale che si riscontra perseguendo questo metodo risolutivo è legato alla fase di creazione del modello della linea, dunque alla traduzione della stessa in un'opportuna catena di Markov.

Più le dimensioni della linea crescono e più sarà elevato il numero di possibili stati del sistema; Inoltre non è da trascurare la difficoltà della risoluzione delle equazioni di stato che portano alla valutazione delle probabilità di stato a regime: infatti accade che per quanto tali equazioni siano lineari, esse divengano sempre più articolate e complesse al crescere della linea e richiedono l'ausilio del calcolatore (e soprattutto di molto tempo) per la loro risoluzione.

Tutto ciò ha portato alla creazione di modelli analitici alternativi, derivati dalla teoria delle code, ma che per mezzo di opportune semplificazioni risultano essere più veloci ed efficaci.

I metodi più importanti fin'ora sviluppati sono sicuramente l'algoritmo di Aggregazione di Dallery e l'algoritmo di Decomposizione di Gershwin.

Ipotesi di base del sistema

Indipendenza degli eventi

L'indipendenza degli eventi implica che lo stato futuro di un sistema risulti essere solamente funzione del suo stato attuale e indipendente da ogni sua altro stato passato. Nei modelli di descrizione di sistemi produttivi, questo significa che il tempo intercorrente fra due rotture e il tempo fra due riparazioni su differenti macchine risultino essere indipendenti fra loro.

Questo inoltre implica che,assumendo che una macchina non sia in stato di "blocking" o "starving", due tempi di rottura successivi su una stessa macchina

risultano essere indipendenti da precedenti fermate, riparazioni e dalla quantità di materiale presente all'interno dei buffer al momento della rottura.
Questa ipotesi permette al sistema di essere modellato come un processo di tipo Markoviano,facilmente analizzabile e trattabile.

Sistema Saturo

Una linea che viene costantemente alimentata dall'esterno si dice satura. Questa ipotesi permette di far si che la prima macchina della linea non possa mai entrare in stato di starving.
Allo stesso modo si assume l' ipotesi che vi sia sempre disponibilità di spazio per accettare il prodotto finito così che l'ultima macchina della linea non possa cadere in stato di blocking.

Politica di Conservazione del lavoro

Questa semplice politica richiede che una macchina non si fermi mai, se non in caso di starving o blocking, nel rispetto della massimizzazione del TP globale.
Tale politica viene utilizzata in quanto rappresenta il sistema di programmazione più usato nella realtà e permette di massimizzare il TP di una linea.

Ipotesi di base sulle macchine

Rotture

Una rottura occorre quando una macchina della linea cessa di lavorare per un malfunzionamento.
Le rotture possono essere del tipo *single stage*,in cui solamente un livello della linea viene interessato o del tipo *multiple stage*, in cui più macchine sono soggette a stop simultaneamente.
Il caso estremo è quello definito di *Total line Failures*, in cui tutte le macchine della linea si trovano in condizione di "shut down" allo stesso momento.
Nella nostra trattazione non considereremo il caso in cui anche i buffer possano essere soggetti a rottura: se infatti considerassimo nastri trasportatori, rulliere ecc.ecc. tali entità risulterebbero essere soggette a rottura come tutte le macchine industriali.
L'assunzione alla base di questa nostra ipotesi è che la frequenza di rottura di tali sistemi sia estremamente ridotta al contrario della frequenza di riparazione. In questo modo tali sistemi possono essere schematizzati come mezzi non soggetti mai a fermate.

Processing Time

L'assunzione che viene fatta è che il tempo ciclo delle macchine considerate sia sempre di tipo deterministico, perché le macchine industriali, in particolare centri di lavoro e macchine a tecnologia CC, presentano tempi di lavorazione contraddistinti da una variabilità estremamente limitata, ipotizzabile dunque come costante.

Non conformità

Con il termine Yield si fa riferimento al numero di parti percentuali conformi della produzione. Solitamente si assume che la produzione sia perfetta e che dunque non siano necessarie rilavorazioni, che andrebbero ad inficiare sul TP globale della linea. Questa ipotesi per quanto non rispecchi la realtà risulta essere necessaria per semplificare i sistema di studio e permettere di schematizzare una linea produttiva come un sistema Markoviano.

Buffer

Buffer è il termine utilizzato per identificare lo spazio in cui vengono stoccate le scorte in-process di una linea. L'ipotesi fondamentale che si utilizza nella loro modellazione è che tali *"storage"* siano sempre di capacità finita.

Con l'avvento del JIT, i buffer di piccole dimensioni sono diventati la norma in molte realtà manifatturiere.

Quando i buffer hanno una capacità ridotta i problemi legati alla rottura di una singola macchina si propagano velocemente sull'intero sistema produttivo.

Buffer di capacità elevate permettono di accogliere un numero molto elevato di pezzi che evitano fenomeni quali lo "Starving" e il "Blocking", danno continuità al flusso della linea (permettono di far si che le macchine collo di bottiglia siano sempre alimentate) e soprattutto disaccoppiano le linee in caso di rotture, alimentandole temporaneamente per mezzo dei semilavorati in essi contenuti.

I buffer sono raramente sovradimensionati per due motivi di carattere economico:

- Lo spazio è una risorsa scarsa e critica negli stabilimenti che solitamente viene allocata per lo svolgimento di attività a valore.
- Più elevata è la capacità di un buffer e maggiore sarà il WIP medio in essi contenuto. Questo si traduce nell'immobilizzo di elevate quantità di capitale.

Buffer Transit Time

Il BTT è il tempo che intercorre dal momento in cui una parte entra in un buffer vuoto non soggetto a fenomeno di blocking dalla linea a monte, fino al tempo in cui la parte può effettivamente essere prelevata per entrare nella linea a valle.
I modelli da noi analizzati e sviluppati assumono sempre BTT nullo: questa ipotesi non è del tutto corretta, in quanto utilizzando sistemi di trasporto come buffers si violerebbe l'ipotesi iniziale, non tanto a regime, quanto in caso di avviamento delle linee post rottura.
L'ipotesi è comunque tollerata in quanto riduce fortemente la complessità di valutazione del sistema.

Zero Buffer Lines

Una linea del tipo Zero Buffer è,come dice il nome stesso, un sistema composto da macchine connesse in serie e non disaccoppiate da opportuni magazzini intermedi. Questa situazione rappresenta la peggiore condizione operativa per una linea in quanto il sistema risulterebbe obbligato a fermarsi completamente ogni volta che su una delle macchine che lo compongono occorresse un guasto. Inoltre non potendo disporre di punti di stoccaggio la linea sarebbe continuamente associata a fenomeni di "Blocking" e "Starving" con nette conseguenze sul TP medio.
Tale condizione non si manifesta mai nella realtà, ma è molto utile avere un'idea di come il TP medio di una linea possa essere influenzato negativamente dall'assenza dei buffer.
Sebbene per linee di tipo sincrono sia stato identificato un modello di valutazione rapido del TP nel caso di buffer nulli (Buzacott), ancora oggi non è stato valutato un sistema rapido e preciso con cui ottenere lo stesso risultato per linee di tipo asincrono e dunque per ottenere tale risultato è necessario fare riferimento alla simulazione o a sistemi di tipo analitico.
Da analisi numeriche è stato rilevato che tale valore può essere anche inferiore del 70% rispetto al caso in cui sulla linea vengano disposti buffer con capacità infinita.

Infinite Buffer Lines

Il caso di linee contraddistinte da buffer dalla capacità infinita è puramente teorico sebbene si possa ricollegare ad uno status in cui su una linea si dispongano buffer dalle capacità molto elevate che a regime possano alimentare per un ampio arco temporale possono sottendere agli effetti di una o più rotture permettendo alla linea di funzionare in maniera continua.

Come è possibile comprendere , una situazione di questo tipo farà si che la Produttività globale della linea sarà data dalla relazione: $TR = min(e_i c_i)$ ovvero tenderà al valore della produttività del bottleneck della linea considerata.

2.2 Metodo di Aggregazione

Il metodo di aggregazione si basa sulle stesse convenzioni attorno a cui è stato sviluppato il modello di valutazione per linee base:

- Sistemi Saturi, dunque flusso di materia prima e richiesta di prodotto finito costante nel tempo e mai nullo.
- Modello a stati indipendenti.
- Modello del tipo Operation-Dependent, un guasto occorre su una macchina se e solo se essa effettivamente funziona ed è operativa.

L'idea alla base dell'aggregazione è piuttosto semplice: considerata una linea di k macchine si aggregano fra di loro in un solo sistema equivalente le ultime macchine che la compongono, ottenendo una linea equivalente, ma di dimensione $(k-1)$.
In maniera ricorsiva continueremo ad aggregare tra loro le ultime due macchine arrivando dunque ad ottenere una macro macchina che rappresenta la linea intera e di cui si possono facilmente rilevare le prestazioni, attraverso le relazioni prima descritte parlando di prestazioni di macchine singole.
Al più è possibile terminare l'aggregazione nel momento in cui si è ricondotti ad una linea base e valutarne le prestazioni per mezzo del modello di Gershwin.
Il procedimento descritto prende nome di Backward Aggregation, in quanto procede dall'ultima macchina fino alla prima, ma logicamente è possibile ottener egli stessi risultati attraverso una propagazione di tipo Forward ovvero aggregando dalla prima all'ultima macchina.
La convergenza dell'algoritmo è molto rapida sebbene questa tecnica pecchi di precisione, soprattutto nel caso in cui si analizzino linee di dimensioni molto elevate, in cui le macchine sono contraddistinte da valori dei parametri caratteristici molto differenti fra di loro.
Comunque la precisione dell'algoritmo diviene sempre maggiore nel momento in cui si descrivano comportamenti asintotici della linea in particolare quando si predispongono buffer dalle capacità molto elevate.

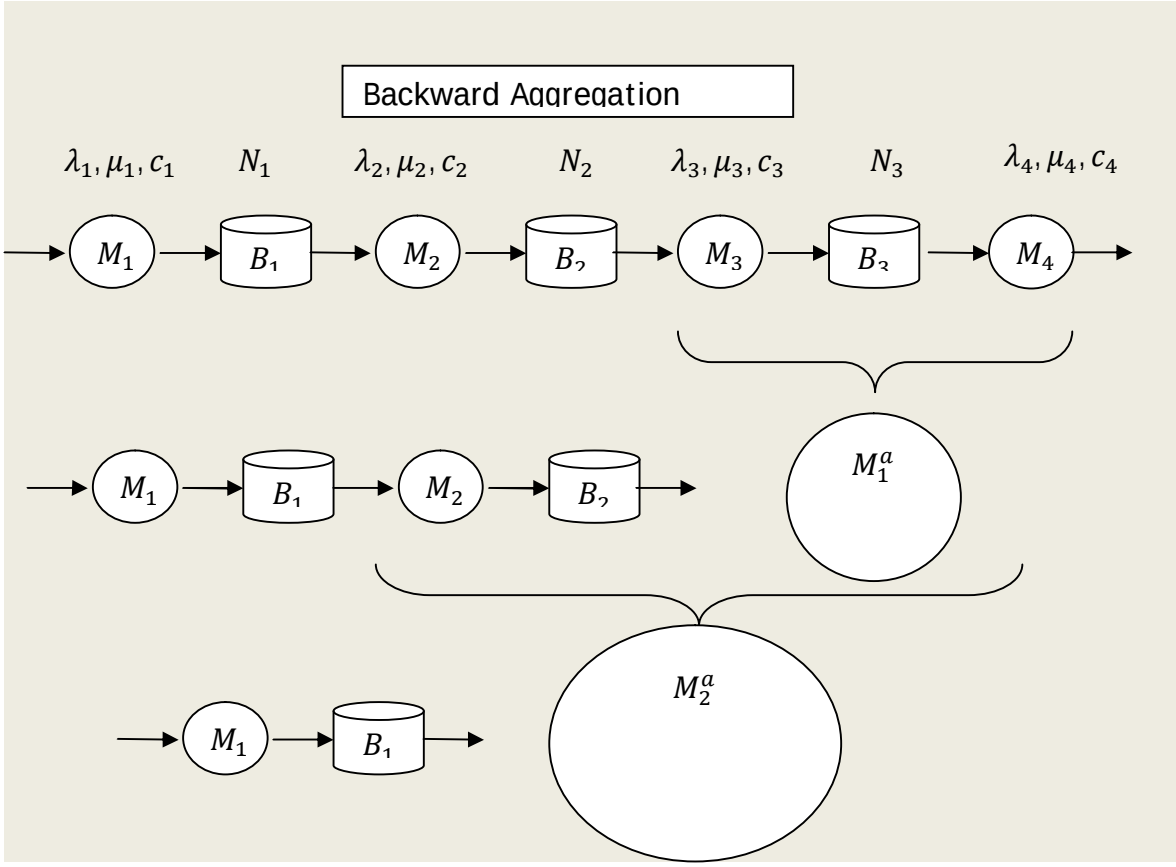


Figura 4, Metodo di Aggregazione

La scarsa precisione di tale sistema di valutazione risiede inoltre nel fatto che nel momento in cui si vanno ad aggregare fra di loro differenti macchine si azzerano tutti i possibili fenomeni di blocking e starvation a cui possono essere soggette le macchine aggregate prese separatamente e dunque non si tiene conto di tutte le perdite di produttività legate a tali fenomeni.
Questa è una delle ragioni principali per cui tale sistema spesso sovrastima le effettive prestazioni di una rete

2.3 Metodo di Decomposizione

Per ovviare al problema dell'imprecisione delle stime legato al metodo di aggregazione si ricorre generalmente ad un metodo di approssimazione più efficace noto come metodo di *decomposizione*.

Tale strumento permette di valutare il Throughput di una linea composta da k macchine, non più riducendola ad una singola macromacchina, bensì decomponendola in un insieme di $(k-1)$ linee che per mezzo di un sistema di omogeneizzazione vengono portate ad avere tutte stesse prestazioni equivalenti a quelle della linea globale.

La generica sottolinea *i*-esima $L(i)$ è composta da due macchine equivalenti $M_u(i)$ e $M_d(i)$ che rappresentano rispettivamente il comportamento aggregato della linea originale a monte (*upstream*) ed a valle (*downstream*) rispetto all'*i*-esimo buffer B_i.

Si noti inoltre che ogni sottolinea presenta un buffer di capacità esattamente uguale all'*i*-esimo buffer della linea originale, logicamente presente nella corrispondente posizione di origine.

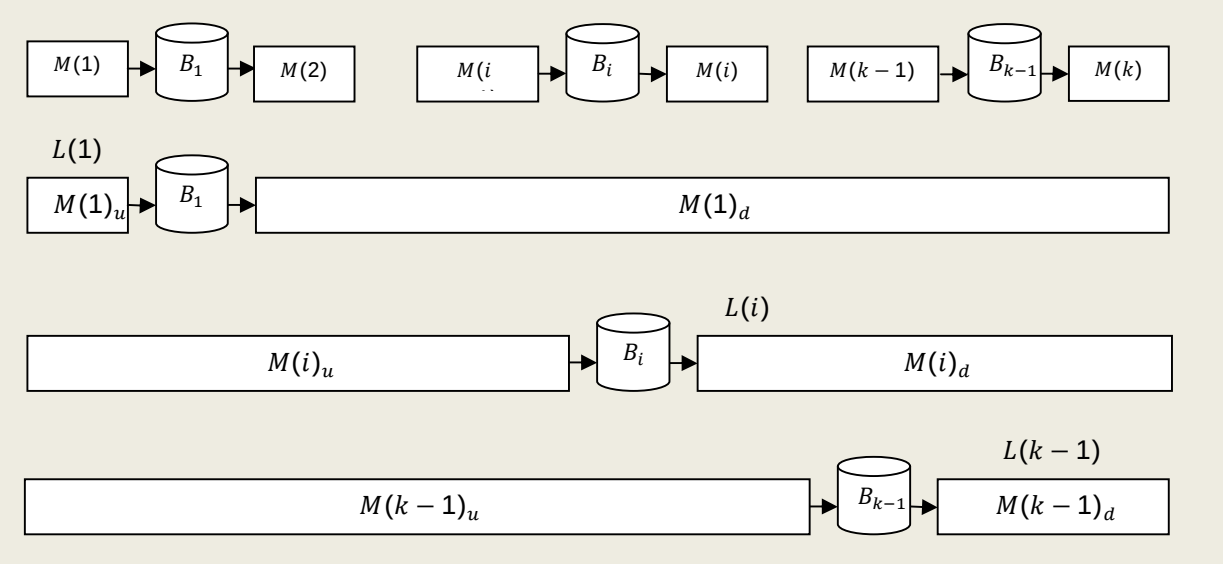


Figura 5, Metodo di Decomposizione

Sebbene possa sembrare che la decomposizione sia simile al sistema di aggregazione, in quanto in ogni sottolinea si aggrega i comportamento delle macchine reali in macchine fittizie, questo sistema di valutazione tenta di tener conto di tutti i fattori che generalmente accadono in una linea come il Blocking e lo Starving.

Per farlo, non condensa tutte le macchine in una sola entità, bensì cerca di "spalmare" i comportamenti tipici su più sottolinee.
Per valutare i parametri delle macchine equivalenti in modo da minimizzare la differenza fra la linea originale e la linea sostitutiva si introducono allora una serie di equazioni di congruenza nel numero di 6(k-1) nel caso di linee asincrone,comuni a tutti i metodi che verranno analizzati di seguito.

Condizioni Generali sulle equazioni di congruenza

Nel seguito forniremo direttamente le equazioni di congruenza che permettono di valutare i parametri descrittivi di ciascuna macchina sostitutiva. Per completezza presentiamo in questo paragrafo le condizioni di partenza che permettono la derivazione delle equazioni di congruenza.
Consideriamo innanzitutto il significato di una fermata della macchina sostitutiva $M_u(i)$. Dato che la $M_u(i)$ schematizza il comportamento della linea a valle rispetto all'i-esimo buffer, uno stop di tale macchina implica che il materiale non fluisce nel buffer B_i a causa di una fermata a monte, per cui devono valere le seguenti condizioni:

$$M_u(i) \quad down = \begin{cases} M_i\ down, & i = 1 \\ M_i\ down\ or\ [M_u(i-1) down\ and\ n_{i-1} = 0), & i = 2, \dots, k-1 \end{cases}$$

$$M_u(i) \quad up = M(i)\ not\ down, \quad i = 1, \dots, k-1$$

Analogamente, nel caso della macchina $M_d(i)$ devono valere le seguenti condizioni:

$$M_d(i) \quad down = \begin{cases} M_{i+1}\ down, & i = k-1 \\ M_{i+1}\ down\ or\ [M_d(i+1) down\ and\ n_{i+1} = N_{i+1}), & i = 1, \dots, k-2 \end{cases}$$

$$M_d(i) \quad up = M(i)\ not\ down, \quad i = 1, \dots, k-1$$

L'efficienza delle macchine $M_u(i)$ e $M_d(i)$ sono rispettivamente $E_u(i)$ e $Ed(i)$, definite come:

$$E_u(i) = prob\{s_u(i) = 1, n_i < N_i) + P_i(n_i = N_i, s_u(i) = 1, s_d(i) = 1) \cdot \frac{c_d(i)}{c_u(i)}$$

$$E_u(i) = prob\{s_d(i) = 1, n_i > 0) + P_i(n_i = 0, s_u(i) = 1, s_d(i) = 1) \cdot \frac{c_u(i)}{c_d(i)}$$

Si noti come il secondo termine delle precedenti equazioni tenga conto della riduzione di efficienza (ovvero all'adeguamento del ritmo alla macchina collo di

bottiglia) quando i buffer intermedi hanno esaurito la loro capacità di disaccoppiare i vari tratti della linea.
Nota l'efficienza, è evidente che il tasso produttivo P delle macchine 'non isolate' debba valere:

$$P_u(i) = E_u(i)c_u(i), \quad i = 1, \dots, k-1$$

$$P_u(i) = E_d(i)c_d(i), \quad i = 1, \dots, k-1$$

E quindi anche il tasso produttivo della generica sotto linea L(*i*) deve valere:

$$P(i) = P_u(i) = P_d(i), \quad i = 1, \dots, k-1$$

Dato che nessuna macchina crea/distrugge materiale, il flusso si deve mantenere costante e quindi, a regime, ogni sotto linea deve essere caratterizzata dallo stesso tasso produttivo P(*i*) = P = $cost$. Questa è la principale condizione al contorno che deve essere soddisfatta dalle equazioni di congruenza.
Vediamo ora più nel dettaglio quali siano le condizioni logiche che permettono di ricavare le 6(k-1) variabili del problema.

Interruptions of Flow Equation:

Per valutare il tasso di guasto delle macchine virtuali si ricorre alle equazioni IOFs (*Interruption of Flow Equations*) che descrivono il comportamento (ovvero l'effetto) di guasto secondo la prospettiva dei buffer.
Le IOFs sono così definite:

$$\lambda_u(i)\delta t + o(\delta t) = probabilità\ che\ M_u(i) si\ guasti\ nell'intervallo$$

$$di\ tempo\ (t, t+\delta t) quando\ n_i < N_i$$

$$\lambda_u(i)\delta t + o(\delta t) = prob[s_u(i) = 0\ in\ t+\delta t | s_u(i) = 1\ and\ n_i < N_i\ in\ t]$$

$$\lambda_d(i)\delta t + o(\delta t) = probabilità\ che\ M_d(i-1) si\ guasti\ nell'intervallo$$

$$di\ tempo\ (t, t+\delta t) quando\ n_{i-1} > 0$$

$$\lambda_d(i-1)\delta t + o(\delta t) = prob[s_d(i-1) = 0\ in\ t+\delta t | s_d(i-1) = 1\ and\ n_{i-1} > 0\ in\ t]$$

Si noti che il termine o(δt) è un infinitesimo di ordine superiore (che verrà trascurato nella trattazione analitica) introdotto esclusivamente per indicare la

possibilità (remota) di eventi rari quali la rottura simultanea di più macchine della linea.

Conoscendo le probabilità di stato a regime della linea equivalente (a due macchine) è possibile utilizzare le IOFs per scrivere un insieme di 2(*k*-2) equazioni di congruenza.

Resumption of Flow Equations:

Analogamente a quanto appena detto, per valutare il tasso di riparazione delle macchine virtuali si ricorre alle equazioni ROFs (*Resumption of Flow Equations*) che descrivono il comportamento (ovvero l'effetto) di un ripristino secondo la prospettiva dei buffer.

Le ROFs sono così definite:

$$\mu_u(i)\delta t + o(\delta t) = probabilità\ che\ M_u(i)\ torni\ a\ funzionare\ nell'intervallo$$

$$di\ tempo\ (t, t + \delta t)$$

$$\mu_u(i)\delta t = prob[s_u(i) = 1\ in\ t + \delta t | s_u(i) = 0\ in\ t]$$

$$\mu_d(i-1)\delta t = probabilità\ che\ M_d(i-1)\ torni\ a\ funzionare\ nell'intervallo$$

$$di\ tempo\ (t, t + \delta t)$$

$$\mu_d(i-1)\delta t = prob[s_d(i-1) = 0\ in\ t + \delta t | s_d(i-1) = 0\ in\ t]$$

Come prima, conoscendo le probabilità di stato a regime della linea equivalente (a due macchine) è possibile utilizzare le IOFs per scrivere un insieme di 2(*k*-2) equazioni di congruenza.

Conservation of Flow Equations

Come precedentemente osservato, la conservazione (ovvero il bilancio) dei flussi di ingresso/uscita lungo la linea produttiva, permette di scrivere altre (*k*-2) equazioni definite COFs (Conservation of Flow Equations). Banalmente, tali equazioni impongono che a regime il throughput debba essere uguale in ogni stazione, dato che lungo la linea 'nulla si crea e nulla si distrugge'.

$$P(i) = P(1), \qquad i = 2, \dots, k-1$$

Flow Rate – Idle Time Equations

Chiaramente la probabilità che una macchina sia *idle* è pari alla somma delle probabilità che la stessa macchina sia *starved* o *blocked*. Tale osservazione permette di scrivere altre (*k*-2) equazioni note come FR-ITs (Flow Rate – Idle Time Equations) e definite nel modo seguente:

$$P(i) = \epsilon_i c_i [1 - p_s(i-1) - p_b(i)], i = 2, \dots, k-1$$

dove:

$\epsilon_i = efficienza\ di\ macchina\ singola\ (a\ regime) =$

$$= \frac{\mu_i}{\lambda_i + \mu_i}, i = 1, \dots, k$$

$p_s(i) = la\ probabilità\ di\ stato\ a\ regime\ che\ l'i-esima\ stazione\ sia\ in\ starving =$

$$= P_{0,0,1}(i), \qquad i = 2, \dots, k$$

$p_d(i) = la\ probabilità\ di\ stato\ a\ regime\ che\ l'i-esima\ stazione\ sia\ in\ blocking =$

$$= p_d(i) = P_{N_i,1,0}(i), i = 1, \dots, k-1$$

Condizioni al contorno

Come detto le IOFs insieme alla COFs ed alle FR-IT forniscono 6(*k*-2) equazioni. Dato che per applicare il metodo di decomposizione servono 6(*k*-1) equazioni è necessario introdurre 6 condizioni al contorno. A tal fine è sufficiente osservare che la macchina $M_u(1)$ e la macchina $M_d(k\text{-}1)$ (ovvero la prima e la seconda macchina delle sottolinee L(1) e L(k-1)) devono coincidere, rispettivamente, con la prima e con l'ultima macchina della linea reale. Devono allora valere le seguenti:

$$\lambda_u(1) = \lambda_1$$

$$\mu_u(1) = \mu_1$$

$$c_u(1) = c_1$$

$$\lambda_d(k-1) = \lambda_k$$

$$\mu_d(k-1) = \mu_k$$

$$c_d(k-1) = c_k$$

Data la loro complessità il sistema delle equazioni di congruenza deve essere risolto in maniera iterativa. La logica su cui si basano i sistemi che andremo a descrivere e utilizzare è la seguente:

- Si fissano le condizioni al contorno;
- Si assegnano valori di partenza ai parametri delle UpStream Machine;
- Si assegnano valori di partenza ai parametri delle DownStream Machines
- Si usano tali valori per stimare i parametri delle DownStream Machines, rispettando i vincoli imposti dalle equazioni di congruenza;
- Si usano i parametri stimati per le DownStream Machinea per ri-stimare i parametri delle upstream machine;
- …..
- Si procede così fino alla convergenza dei valori e la successiva valutazione delle prestazioni globali della linea.

Dunque a differenza del metodo di aggregazione qui si creano (k-1) linee M2 equivalenti, che nel momento in cui convergono possono essere studiate semplicemente attraverso il modello di Gershwin.

Il fatto di aver creato (k-1) sottolinee permette di mantenere l'effetto dei fenomeni di Blocking e Starving a cui sono soggette le singole macchine fino alla convergenza dell'algoritmo e dunque ottenere risultati soddisfacenti con una precisione più elevata

2.4 DDX (Dallery-David-Xie Algorithm).

Il DDX, sviluppato nel 1988 dal francese Dallery, per più di un decennio ha rappresentato l'algoritmo di riferimento per la valutazione delle prestazioni di linee asincrone composte da macchine a tre parametri.
Il suo punto di forza è sicuramente una buona capacità di convergenza e l'applicabilità ad un parco di casi esteso, ma pecca per velocità, semplicità computazionale e precisione.

Equazioni di Riferimento

Sviluppando le generiche equazioni prima elencate e dopo alcuni passaggi algebrici si ottengono le seguenti equazioni specifiche per la valutazione delle linee equivalenti di monte e di valle:

Upstream Equations

$$\lambda_u(i) = \lambda_i\left(1 + \frac{P_{0,1,1}(i-1)c_u(i)}{TR(i) - P_{N_i,1,1}(i)c_d(i)} \cdot \left(\frac{c_u(i-1)}{c_i} - 1\right)\right) + \left(\frac{P_{0,0,1}(i-1)c_u(i)}{TR(i) - P_{N_i,1,1}(i)c_d(i)}\right)\mu_u(i-1), \quad i = 2, \dots, k-1 \qquad (3.7)$$

$$\mu_u(i) = \mu_u(i-1)\frac{P_{0,0,1}(i-1)\mu_u(i)c_u(i)}{\lambda_u(i)TR(i)} + \mu_i\left(1 - \frac{P_{0,0,1}(i-1)\mu_u(i)c_u(i)}{\lambda_u(i)TR(i)}\right), \quad i = 2, \dots, k-1 \qquad (3.8)$$

$$c_u(i) = \frac{1}{\epsilon_u(i)}\left\{\frac{1}{\frac{1}{TR(i)} + \frac{1}{\epsilon_i c_i} - \frac{1}{\epsilon_d(i-1)c_d(i-1)}}\right\}, i = 2, \dots, k-1 \qquad (3.9)$$

Downstream Equations

$$\lambda_d(i) = \lambda_{i+1}\left(1 + \frac{P_{N_{i+1},1,1}(i+1)c_d(i)}{TR(i) - P_{0,1,1}(i)c_u(i)} \cdot \left(\frac{c_d(i+1)}{c_{i+1}} - 1\right)\right) + \left(\frac{P_{N_{i+1},1,0}(i+1)c_d(i+1)}{TR(i) - P_{0,1,1}(i)c_u(i)}\right)\mu_d(i+1), \quad i = 1, \dots, k-2 \qquad (3.10)$$

$$\mu_d(i) = \mu_d(i+1)\frac{P_{N_{i+1},1,0}(i+1)\mu_d(i)c_d(i)}{\lambda_d(i)TR(i)} + \mu_{i+1}\left(1 - \frac{P_{N_{i+1},1,0}(i+1)\mu_d(i)c_d(i)}{\lambda_d(i)TR(i)}\right),$$
$$i = 1, \dots, k-2 \qquad (3.11)$$

$$c_d(i) = \frac{1}{\epsilon_d(i)}\left\{\frac{1}{\frac{1}{TR(i)} + \frac{1}{\epsilon_{i+1}c_{i+1}} - \frac{1}{\epsilon_u(i+1)c_u(i+1)}}\right\}, i = 1, \dots, k-2 \qquad (3.12)$$

Boundary Conditions

$$\lambda_u(1) = \lambda_1$$

$$\mu_u(1) = \mu_1$$

$$\mu_c(1) = c_1$$

$$\lambda_d(k-1) = \lambda_k$$

$$\mu_d(k-1) = \mu_k$$

$$c_d(k-1) = c_k$$

Algoritmo risolutivo

Step 1: *Inizializzazione*

Si ponga:

$$\lambda_u(1) = \lambda_1;\ \mu_u(1) = \mu_1;\ c_u(1) = c_1;$$

$$\lambda_d(i) = \lambda_{i+1};\ \mu_d(i) = \mu_{d+1};\ c_d(i) = c_{d+1};$$

...

$$\lambda_d(k-1) = \lambda_k;\ \mu_d(k-1) = \mu_k;\ ;\ c_d(i) = c_d(k-1);$$

Step 2: *parametri delle 'UpStream Machine' calcolati in ordine progressivo*

Per $i = 2, \dots, (k-1)$ si calcolino le probabilità di stato di ogni sottolinea $L(i)$ e, in base a tali probabilità $Pzjk(i)$, si valutino nel seguente ordine i tre parametri di ogni UpStream machine.

$$\lambda_u(i) = \lambda_i\left(1 + \frac{P_{0,1,1}(i-1)c_u(i)}{TR(i-1)} \cdot \left(\frac{c_u(i-1)}{c_d(i-1)} - 1\right)\right) + \left(\frac{P_{0,0,1}(i-1)c_u(i)}{TR(i-1)}\right)\mu_u(i-1), \quad (3.13)$$

$$\mu_u(i) = \mu_u(i-1)\frac{P_{0,0,1}(i-1)\mu_u(i)c_u(i)}{\lambda_u(i)TR(i-1)} + \mu_i\left(1 - \frac{P_{0,0,1}(i-1)\mu_u(i)c_u(i)}{\lambda_u(i)TR(i-1)}\right) \quad (3.14)$$

$$c_u(i) = \frac{1}{\epsilon_u(i)}\left\{\frac{1}{\frac{1}{TR(i-1)} + \frac{1}{\epsilon_i c_i} - \frac{1}{\epsilon_d(i-1)c_d(i-1)}}\right\} \quad (3.15)$$

Dove i valori di $P_{0,1,1}$, $P_{0,0,1}$, TR si calcolano rispettivamente mediante le (1.17 a-b), (1.19 a-b) e (1.26), ovvero le equazioni del modello di valutazione delle linee di base.

Step 3: *parametri delle 'DowStream Machines' calcolati in ordine inverso*
Per i = 1,….,(k-2) si calcolino le probabilità di stato di ogni sottolinea L(i) e, in base a tali probabilità $P_{zjk}(i)$, si valutino nel seguente ordine i tre parametri di ogni downstream machine:

$$\lambda_d(i) = \lambda_{i+1}\left(1 + \frac{P_{N_{i+1},1,1}(i+1)c_d(i)}{TR(i+1)} \cdot \left(\frac{c_d(i+1)}{c_u(i+1)} - 1\right)\right) + \left(\frac{P_{N_{i+1},1,0}(i+1)c_d(i+1)}{TR(i+1)}\right)\mu_d(i+1), \quad (3.16)$$

$$\mu_d(i) = \mu_d(i+1)\frac{P_{N_{i+1},1,0}(i+1)\mu_d(i)c_d(i)}{\lambda_d(i)TR(i+1)} + \mu_{i+1}\left(1 - \frac{P_{N_{i+1},1,0}(i+1)\mu_d(i)c_d(i)}{\lambda_d(i)TR(i+1)}\right), \quad (3.17)$$

$$c_d(i) = \frac{1}{\epsilon_d(i)}\left\{\frac{1}{\frac{1}{TR(i+1)} + \frac{1}{\epsilon_{i+1}c_{i+1}} - \frac{1}{\epsilon_u(i+1)c_u(i+1)}}\right\} \quad (3.18)$$

Dove i valori di $P_{0,1,1}$, $P_{0,0,1}$, TR si calcolano rispettivamente mediante le (1.17 a-b), (1.19 a-b) e (1.26).

Step 4: *terminazione*
Si ripetano gli step 2-3 fino a quando

$$\|P(i) - P(1)\| \leq \varepsilon \;\; \forall\, i = 2, \dots, k-1$$

2.4.1 Improved DDX

Come è facilmente rilevabile il DDX presenta una complessità computazionale intrinseca, legata al fatto che ad ogni step di valutazione delle linee equivalenti di valle e di monte, questo richieda la risoluzione di un sistema di tre equazioni non lineari in tre incognite.

Tale problema è stato eliminato attraverso una semplice analisi di tali equazioni che con alcuni passaggi algebrici ed opportune sostituzioni sono state ridotte ad equazioni lineari ad unica incognita.

Il risultato finale di questa prima fase di miglioramento è sintetizzata nelle seguenti equazioni:

UpStream Equation

$$\mu_u(i) = \left\{[(Y-L)K + \mu_i] + (L-Y)\left(\frac{(T+Y)K}{F}\right)K\right\}\left[\left(\frac{F}{F-(Y-L)K}\right)\right]$$

$$\lambda_u(i) = (\lambda_i + (T+Y)K)\left(\frac{\mu_u(i)}{\mu_u(i)-(T+Y)K}\right) = F\left(\frac{\mu_u(i)}{\mu_u(i)-(T+Y)K}\right)$$

$$c_u(i) = \left(\frac{\lambda_u(i)+\mu_u(i)}{\mu_u(i)}\right)K = \left(\frac{\lambda_u(i)}{\mu_u(i)}+1\right)K$$

Parametri Sostituitivi per equazioni di UpStream

$$\alpha = \left(\frac{P_{0,1,1}(i-1)}{TR(i-1)}\right), \beta = \left(\frac{c_u(i-1)}{c_d(i-1)}-1\right), \kappa = \left\{\frac{1}{\frac{1}{TR(i-1)}+\frac{1}{\epsilon_i c_i}-\frac{1}{\epsilon_d(i-1)c_d(i-1)}}\right\},$$

$$\gamma = \left(\frac{P_{0,0,1}(i-1)}{TR(i-1)}\right), Y = \gamma\mu_u(i-1), T = \left(\lambda_i(\alpha\beta)\right), L = \mu_i(\gamma)$$

$$F = (\lambda_i + (T+Y)K)$$

DownStream Equation

$$\mu_d(i) = \left\{[(\Upsilon-l)\Theta + \mu_{i+1}] + (l-\Upsilon)\left(\frac{(\tau+\Upsilon)\Theta}{f}\right)\Theta\right\}\left[\left(\frac{f}{f-(\Upsilon-l)\Theta}\right)\right]$$

$$\lambda_d(i) = f\left(\frac{\mu_d(i)}{\mu_d(i)-(\tau+\Upsilon)\Theta}\right)$$

$$c_d(i) = \left(\frac{\lambda_d(i)}{\mu_d(i)}+1\right)\Theta$$

Parametri Sostituitivi per equazioni di DownStream

$$\mathrm{A} = \left(\frac{P_{N_{i+1},1,1}(i+1)}{TR(i+1)}\right), \mathrm{B} = \left(\frac{c_d(i+1)}{c_u(i+1)} - 1\right), \Theta = \left\{\frac{1}{\frac{1}{TR(i+1)} + \frac{1}{\epsilon_{i+1}c_{i+1}} - \frac{1}{\epsilon_u(i+1)c_u(i+1)}}\right\},$$

$$\Delta = \left(\frac{P_{N_{i+1},1,0}(i+1)}{TR(i+1)}\right), \Upsilon = \Delta\mu_d(i+1), \qquad , \tau = (\lambda_{i+1}AB), l = \mu_{i+1}\Delta$$

$$f = (\lambda_{i+1} + (\tau + \Upsilon)\Theta)$$

Come è perfettamente comprensibile, le modifiche apportate non comportano un incremento dell'efficacia dell'algoritmo (dunque la sua frequenza di convergenza e la sua capacità di fornire risultati vicini alla realtà), bensì ne incrementa solo l'efficienza ovvero la velocità computazionale.

2.5 ADDX (Accelerated DDX)

L'ADDX rappresenta la vera evoluzione del DDX precedentemente trattato e nasce dall'esigenza di disporre di un algoritmo di valutazione più efficace ed efficiente, che permetta di studiare un ampia gamma di casi differenti, in maniera pratica e veloce.

Burman,suo ideatore e sviluppatore, è partito nella sua ricerca dal DDX di Dallery a cui ha apportato opportune modifiche per ottenere i seguenti risultati:

- Passare da un set risolutivo di equazioni non lineari ad un set di equazioni lineari (risultato ottenuto precedentemente anche nella nostra trattazione).
- Rendere l'algoritmo più efficace e fornirgli una frequenza di convergenza più elevata.

Il primo obiettivo è stato raggiunto ponendo le equazioni di UpStream nella seguente forma:

$$\lambda_u(i) = c_u(i)K_1 + \mu_u(i)$$

$$\mu_u(i) = \frac{\mu_u(i)}{\lambda_u(i)} c_u(i)K_2 + \mu_i$$

$$c_u(i) = \frac{\lambda_u(i)}{\mu_u(i)} K_3 + K_3$$

Allo stesso modo è possibile identificare una forma sintetica per le equazioni di DownStream, che una volta trattate algebricamente portano alle seguenti equazioni sostitutive di (3.13) – (3.18)

$$\lambda_u(i) = \frac{\lambda_i K_2 K_3 + \mu_i \lambda_i + \mu_i K_1 K_3}{\mu_i + K_2 K_3 - K_1 K_3}$$

$$\mu_u(i) = \frac{\lambda_i K_2 K_3 + \mu_i \lambda_i + \mu_i K_1 K_3}{\lambda_i + K_1 K_3 - K_2 K_3}$$

$$c_u(i) = \frac{K_3(\mu_i + \lambda_i)}{\mu_i + K_2 K_3 - K_1 K_3}$$

$$\lambda_d(i) = \frac{\lambda_{i+1} K_5 K_6 + \mu_{i+1} \lambda_{i+1} + \mu_{i+1} K_4 K_6}{\mu_{i+1} + K_5 K_6 - K_4 K_6}$$

$$\mu_d(i) = \frac{\lambda_{i+1} K_5 K_6 + \mu_{i+1} \lambda_{i+1} + \mu_{i+1} K_4 K_6}{\lambda_{i+1} + K_4 K_6 - K_5 K_6}$$

$$c_d(i) = \frac{K_6(\lambda_{i+1} + \mu_{i+1})}{\mu_{i+1} + K_5 K_6 - K_4 K_6}$$

Dove:

$$K_1 = \lambda_i \left(\frac{P_{0,1,1}(i-1)}{PR(i-1)} \left(\frac{c_u(i-1)}{c_d(i-1)} - 1 \right) \right) + \left(\frac{P_{0,0,1}(i-1)}{PR(i-1)} \right) \mu_u(i-1)$$

$$K_2 = (\mu_u(i-1) - \mu_i) \left(\frac{P_{0,0,1}(i-1)}{PR(i-1)} \right)$$

$$K_3 = \frac{1}{\frac{1}{TR(i-1)} + \frac{1}{\epsilon_i c_i} - \frac{1}{\epsilon_d(i-1) c_d(i-1)}}$$

$$K_4 = \lambda_{i+1} \left(\frac{P_{N_{i+1},1,1}(i+1)}{PR(i+1)} \left(\frac{c_d(i+1)}{c_u(i+1)} - 1 \right) \right) + \left(\frac{P_{N_{i+1},1,0}(i+1)}{PR(i+1)} \right) \mu_d(i+1)$$

$$K_5 = (\mu_d(i+1) - \mu_{i+1}) \left(\frac{P_{N_{i+1},1,0}(i+1)}{PR(i+1)} \right)$$

$$K_6 = \frac{1}{\frac{1}{TR(i+1)} + \frac{1}{\epsilon_{i+1} c_{i+1}} - \frac{1}{\epsilon_u(i+1) c_u(i+1)}}$$

Per far si che l'algoritmo risolutivo converga con maggiore frequenza è necessario apportare alcune piccole modifiche alle equazioni precedentemente viste :

$$c_i \Rightarrow c_d(i-1)$$

$$TR(i) - P_{N_i,1,1}(i) c_d(i) \Rightarrow TR(i)$$

$$c_{i+2} \Rightarrow c_d(i+1)$$

$$TR(i) - P_{0,1,1}(i) c_u(i) \Rightarrow TR(i)$$

2.5.1 Analisi Critica del Modello

L'ADDX rappresenta in assoluto l'algoritmo di valutazione di linee produttive asincrone più efficiente e come mostrato dallo stesso Burman, esso possiede una frequenza di convergenza assai più elevata rispetto al DDX classico.
Quando è stato proposto nel 1999, esso è stato validato prendendo a riferimento i casi proposti da Glass e Hong nel 1996, per testare il GH, un algoritmo molto simile per funzionamento all'ADDX stesso.
Dalla sperimentazione Burman riuscì a dimostrare che l'ADDX disponeva di una frequenza di convergenza molto prossima al 100%.
Incoraggiati da tali risultati è stato deciso di utilizzare l'ADDX come pilastro del nostro sistema di ottimizzazione, ma prima di procedere è stato deciso di testare ulteriormente l'algoritmo su un set di casi più ampio, al fine di comprenderne le effettive capacità e stimarne con correttezza l'affidabilità.
Per farlo è stato sviluppato un software pilota, che permette di rendere attivo l'algoritmo: tale software è stato sviluppato piuttosto semplicemente grazie al fatto di avere a disposizione il codice precedentemente implementato per testare il modello di Gershwin.
La decisione di validare ulteriormente l'algoritmo è scaturita dal fatto che gli esempi su cui l'ADDX è stato testato originariamente non sono stati ritenuti sufficientemente completi e perciò è stato ritenuto fondamentale testare il sistema di valutazione mettendolo alla prova su un insieme di casi più completo e vario.
Le critiche mosse ai test effettuati da Burman sono principalmente tre:

- L'ADDX è stato testato su linee di dimensioni molto ridotte (k=3),composte da macchine con tempi ciclo ed efficienze molto simili tra di loro,dunque molto più simili a linee di tipo sincrono che linee di tipo asincrono.
- Gli esempi proposti da Glass e Hong prevedono la valutazione delle linee su valori di capacità dei buffer estremamente elevate, dunque analizzano le stesse solamente a livello asintotico, tralasciando il comportamento dell'ADDX per buffer estremamente ridotti e per valori di capacità produttiva intermedi o misti.
- Le linee random testate sono state generate in modo tale da essere molto bilanciate e simil-omogenee.

2.5.2 Validazione ADDX

Dall'analisi precedentemente condotta emerge l'esigenza di verificare il comportamento dell'ADDX su casi di maggiore complessità, in modo tale da valutarne le effettive prestazioni,per poi poterne realizzare possibili miglioramenti nel caso in cui questo venga ritenuto necessario.
Per rendere il più efficace possibile la nostra ricerca è stato deciso di pianificare gli esperimenti da condurre attraverso una fase di DOE(Design of Experiments), in modo tale da identificare in maniera univoca gli obiettivi da raggiungere con i nostri esperimenti e tracciare chiaramente il sentiero da percorrere durante il loro svolgimento.
Gli obiettivi preposti sono i seguenti:

- Testare il comportamento dell'ADDX su linee di dimensioni elevate.
- Testare il comportamento dell'ADDX su linee composte da macchine molto differenti fra di loro.
- Testare il comportamento dell'ADDX, non solo rispetto a valori asintotici dei buffer, bensì anche utilizzando capacità dei magazzini intermedi limitate.

Per ottenere questi obiettivi sono state identificate le seguenti vie:

- Utilizzare composte da un numero di macchine maggiore di 3
- Per ogni dimensione considerata, usare esempi il più vario possibile, in modo tale da tenere in considerazioni sia linee simil-sincrone, sia linee fortemente sbilanciate.
- Per ogni esempio analizzato realizzare esperimenti con capacità dei buffer di classi differenti, in modo tale da testare l'algoritmo su scale differenti.

Analisi dei dati

Di seguito proporremo in maniera puntuale tutti i risultati dei test effettuati sulla linea 1, caso pilota, utilizzato per valutare in maniera trasversale l'ADDX classico e le sue versioni successive in cui verranno implementati dei miglioramenti da noi suggeriti.
I risultati legati ai rimanenti casi studiati verranno indicati in maniera sintetica al termine del capitolo.
Tutte le linee in questione sono state testate su tre differenti situazioni:

1. Capacità dei buffer ridotte, ognuna delle quali varia dal valore iniziale di 0 al valore max di 5, per valutare la frequenza di convergenza dell'algoritmo su una situazione asintotica legata alla assenza di buffer.
2. Capacità dei buffer intermedia, variabile da un valore iniziale di 10 fino a d un valore superiore di 30.
3. Capacità molto elevate, per testare frequenza di convergenza su valori asintotici dell'algoritmo (buffer di capacità superiore ai 100 elementi).

In ultimo luogo,prima di mostrare i risultati è necessario mettere in evidenza una classificazione da noi postulata dei casi testati:

- **Non Convergenti:**tutti quei casi in cui l'algoritmo non riuscendo a convergere genera dei valori delle variabili computazionali di Overflow ,che portano allo stop del programma.
- **Loop:** sono tutti quei casi in cui l'algoritmo tende a generare ciclicamente i soliti valori. La terminazione nello studio del caso avviene nel momento in cui viene soddisfatta un'opportuna condizione sul numero di iterazioni totali.
- **Convergenti:**tutti quei casi in cui l'algoritmo riesce e a fornire come output un effettivo valore di TP della linea testata.

Risultati Esperimento 1

Linea 1	M1	M2	M3	M4	M5
λ	0.1	0.2	0.1	0.2	0.3
μ	0.9	0.6	0.8	0.9	0.7
c	5	5.3	4.9	5.2	4.7

L'esperimento 1 è stato realizzato due volte, utilizzando valori di soglia per la terminazione dell'algoritmo differenti.
Il motivo per cui è stato deciso di replicare l'esperimento è legato alla volontà di vedere quanto incide il valore prestabilito sulla sua capacità di convergenza.

Caso1.1.1:

Caso 1.1	
Test Effettuati	1296
Non Convergenti	37
Loop	556
Convergenti	703
η	0,54
ε	0.1
Iterazioni Max	30

Caso 1.1.2

Caso 1.2	
Casi Testati	1296
Test Effettuati	26
Loop	464
Convergenti	832
η	0,64
ε	0.25
Iterazioni Max	30

Analisi critica dei risultati

I primi due esperimenti ci mostrano come il comportamento dell'ADDX nel caso in cui si lavori con buffer di dimensioni limitate sia poco efficace. In particolare si può notare come imponendo una soglia di ϵ = 0,25 (ovvero uno scarto assoluto di 0,5 rispetto al TP della prima macchina) si abbia una probabilità di convergenza del 64%.

L'aspetto positivo che si può riscontrare è che i casi in cui l'ADDX diverge in maniera netta, creando problemi di esecuzione rappresentano una quota piuttosto limitata rispetto al totale (circa un 2%), dunque esistono elevati margini di miglioramento.

L'analisi del caso condotto ci ha spinto a studiare in maniera approfondita i motivi per cui si è riscontrato un così elevato tasso di non convergenza dell'algoritmo. Per farlo sono stati isolati alcuni casi pratici di deficit dell'algoritmo per tentare di classificare i problemi del sistema e per ogni categoria di mancanza rilevata definirne le cause.

Divergenza completa:

La divergenza completa si manifesta quando i valori dei TP_i delle $(k-1)$ sottolinee risultano essere troppo differenti fra di loro. Tutto ciò implica che l'effetto di omogeneizzazione che avviene tramite l'aggiornamento dei parametri imposto dall'ADDX diviene poco efficace e il sistema diverge

Dagli esempi 1.1 e 1.2 si riporta il caso B: [2,0,20] che mostra questo fenomeno:

Step	TR 1	TR 2	TR 3	TR 4
1	3.2248	2.4123	2,1123	2.0437
2	3,1891	2,4006	2,0364	1,9976

Lo step 3 comporta la terminazione dell'algoritmo per "Floating Point Overflow" (incapacità del software di eseguire una determinata operazione, come una divisione per un numero troppo piccolo o il calcolo di un esponenziale con un esponente molto elevato), proprio a causa del fatto che al passo 2, il TP_1 assume un valore molto più elevato rispetto ai rimanenti TP_i delle sottolinee successive.

Loop Simmetrico

Quando l'algoritmo entra in Loop simmetrico accade che dopo alcuni passi in cui i TP_i delle sottolinee tendono a convergere fra di loro, essi si stabiliscano su una situazione stazionaria.

Dagli esempi 1.1 e 1.2 si riporta il caso B: [1,4,0,0] che mostra questo comportamento:

Step	TR 1	TR 2	TR 3	TR 4
1	3,3075	2,6377	2,5021	2,4757
2	2,6583	2,1590	2,4382	2,4461
2	1,961	1,97	2,48	2,48
3	2,097	2,07	2,45	2,45
4	2,0610	2,0706	2,4522	2,4522
5	2,0707	2,0707	2,4553	2,4553
…	…	…	…	…
…	…	…	…	…
…	…	…	…	…
N	2,0707	2,0707	2,4553	2,4553

In questo caso si può notare come al passo 2 l'aggiornamento delle prime due sottolinee sia stato eccessivo e dunque i valori di riferimento siano stati "travisati" generando una sorta di scoglio, difficilmente oltrepassabile dall'algoritmo nella fase in cui tenta di omogeneizzare i TP_i.

Loop Asimmettrico

Anche nel caso del loop asimmetrico si riscontra che l'algoritmo converge molto velocemente, ma dopo aver raggiunto il punto di massima convergenza questo si discosta completamente dalla soluzione ottimale e entra in loop.

B: [1,5,0,0]

Step	TR 1	TR 2	TR 3	TR 4
1	3,3	2,63	2,6	2,56
2	2,658	2,2577	2,5398	2,54833
2	2,06	0,401	0,299	0,375
3	1,6	0,95522	0,7915	0,902
…	0,4029	1,65	2,41	2,29
…	1,6	0,95522	0,7915	0,902
N	0,4029	1,65	2,41	2,29

Anche in questo caso si può osservare come l'algoritmo non abbia raggiunto la convergenza a causa di una differenza troppo elevata fra il TP_1 e ilTP_2,valore anomalo generato dallo squilibrio di buffer fra la macchina 2 e la macchina 5 al passo 2

Caso 1.2

Linea 1.2	
Test Effettuati	1296
Non Convergenti	62
Loop	802
Convergenti	402
η	0,31
ε	0.25
Iterazioni Max	30

Dalla'analisi condotta sui valori intermedi (da 5 a 9) si è riscontrato un peggioramento delle prestazioni a livello di convergenza, ma allo stesso tempo è stato rilevato un fatto incoraggiante:i casi di Loop sono ancora più convergenti rispetto a quelli trattati precedentemente.

Ad esempio, imponendo il buffer B: [8,7,8,6] s ottiene il seguente risultato:

Step	TR 1	TR 2	TR 3	TR 4
1	3,49	3,16	3,15	3,13
2	3,22	2,85	3,14	3,08
2	2,76	3,033	3,22	3,08
3	2,85	3,033	3,22	3,08
…	2,76	3,033	3,22	3,08
…	2,85	3,033	3,22	3,08
N	2,76	3,033	3,22	3,08

In questo caso è evidente il fatto che l'algoritmo abbia trovato la "via" della convergenza, ma si blocchi a causa di una difficoltà a soddisfare la condizione di stop.

Tutto ciò presuppone che con delle opportune modifiche sarà possibile trasformare tali casi in casi convergenti.

Caso 1.3-→valori elevati

Caso 1.3	
Test Effettuati	1296
Non Convergenti	91
Loop	0
Convergenti	1205
η	0,93
ε	0.2
Iterazioni Max	30

Ciò che si riscontra è che per valori molto elevati l'algoritmo non va mai in loop, sebbene aumentino in maniera consistente i casi che non possono essere valutati. Dall'analisi di tali casi è riscontrato che essi convergono in maniera molto rapida (nel 98% dei casi già alla prima chiamata dell'algoritmo), per poi esplodere alla chiamata successiva

La tabella successiva mostra i risultati ottenuti sugli stessi valori di buffer con i seguenti esempi:

Linea	**2.1**	**2.2**	**2.3**	**3.1**	**3.2**	**3.3**	**4.1**	**4.2**	**4.3**
Test Effettuati	1296	1296	1296	1296	1296	1296	1296	1296	1296
Non Convergenti	37	1	0	12	0	0	42	1	0
Loop	556	530	0	62	6	0	602	431	0
Convergenti	703	765	1296	1222	1290	1296	652	864	1296
η	0,54	0,59	100	0,94	0,99	1	0,5	0,66	100
ε	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
Iteraz. Max	30	30	30	30	30	30	30	30	30

Un'analisi consuntiva dei risultati ci permette di definire le seguenti caratteristiche dell'ADDX originale:

- Più la linea analizzata è simil-omogenea e più la frequenza di convergenza è elevata.
- L'algoritmo ha una scarsa frequenza di convergenza quando si hanno tempi di fermo medi nettamente superiori rispetto ai tempi ciclo delle macchine che compongono la linea.
- Quando la linea è fortemente non omogenea l'algoritmo tende a convergere raramente se fra le macchine sono presenti buffer con capacità nulla, perché tali situazioni divengono difficilmente valutabili.

2.5.3 1° Miglioramento: Modifica del criterio di convergenza.

Il criterio di convergenza dell'ADDX, ereditato in toto dal DDX di Dallery, impone che l'algoritmo termini se:

$$\max(|TP_1 - TP_i|) < con\,\varepsilon\; i{:}2..(k-1)$$

Questo presuppone dunque che i parametri delle differenti sottolinee vengano aggiornati in modo tale che si possano omogeneizzare in maniera completa e uniforme e che dunque siano contraddistinte tutte dallo stesso Throughput.
Negli esempi sviluppati nella prima parte della trattazione abbiamo usato valori di ε differenti, per vedere come variasse la probabilità di convergenza dell'algoritmo dando limiti meno stringenti al criterio di convergenza (dunque ampliando lo spazio delle possibili soluzioni).
E' stato riscontrato che spesso, anche impostando valori di ε elevati (ad esempio imponendo margini maggiori del 10% rispetto al valore della soluzione cercata), l'ADDX non presenta una frequenza di convergenza elevata, in quanto afflitto da numerosi casi di Overflow e di Loop.
Proprio dall'analisi puntuale dell'evolversi dei TP_idi questi ultimi è nata l'idea di modificare il criterio di stop: spesso infatti accade che una volta raggiunto il picco di convergenza, il criterio in questione non venga soddisfatto, portando al collasso l'algoritmo stesso, sebbene sia chiaro che i differenti TP stiano uniformandosi su uno stesso valore.
In particolare è stato evidenziato come spesso la causa alla base di questo problema è la presenza di un'unica sottolinea che non è riuscita a convergere e dunque a omogeneizzarsi con il resto del sistema.
La tabella riporta due esempi tratti dal caso 3.1, in cui in rosso sono stati identificati i parametri non convergenti.

Esempio	TR_1	TR_2	TR_3	TR_4
0-1-5-2	4.82	4.658	4.660	4.661
1-0-1-2	4.288	4.445	4.312	4.288

Nel primo esempio sembra chiaro come il TP globale della linea debba attestarsi intorno al valore 4,66 così come nel secondo esempio sembra chiaro che il valore di convergenza debba essere ricadere nell'intorno del valore 4,3.

L'idea di fondo è quella di valutare la media dei valori dei TP_i, che chiameremo $TP_M = \frac{\sum_1^{k-1} TP_i}{(k-1)}$, per ogni ciclo duplice di aggiornamento (up/downstream), per poi confrontarlo con i TP_i delle sottolinee, in modo tale da sfruttare al meglio il criterio di convergenza prima imposto.

Il presupposto alla base di tale modifica è che se la maggioranza delle sottolinee di decomposizione ha raggiunto un buon livello di convergenza, gli errori dovuti alla presenza di sottolinee meno convergenti possono essere così inglobato nella computazione globale e dunque divenire minimo. L'effetto di tale modifica diviene ancora più consistente nel momento in cui tale fenomeno si manifesti su linee di dimensioni molto elevate (ad esempio 10/15 macchine) in cui tale attività di "suddivisone dell'errore" diviene ancora più efficace.

Dunque imporremo che l'algoritmo si fermi quando:

$$\max(|TP_M - TP_i|) < con\ \varepsilon\ i: 1..(k-1)$$

Tale pratica, non influisce sulla valutazione dei casi già convergenti, mentre permette di "recuperare", buona parte di quei casi che inizialmente cadevano in Overflow e in Loop.

Logicamente il valore del Throughput globale restituito sarà quello del TP_M e non più quello della TP_1 come in precedenza.

Il nuovo criterio di convergenza è stato dunque testato sui casi prima descritti.

Caso 1.1

Linea 1	M1	M2	M3	M4	M5
λ	0.1	0.2	0.1	0.2	0.3
μ	0.9	0.6	0.8	0.9	0.7
c	5	5.3	4.9	5.2	4.7

Caso 1 Modifica		***Caso 1.2***	
Test Effettuati	1296	Test Effettuati	1296
Non Convergenti	11	Non Convergenti	62
Loop	351	Loop	802
Convergenti	934	Convergenti	402
η	0,72	η	0,31
ε	0.20	ε	0.25
Iterazioni Max	30	Iterazioni Max	30

La linea 1 è stata presa a riferimento perché rappresenta in assoluto il caso su cui l'ADDX ha trovato maggiori difficoltà a convergere.
La tabella sopra riportata permette di confrontare le prestazioni dell'algoritmo a cui è stata aggiunta la modifica e il caso 1.2.
Il caso 1.2 è stato scelto per evidenziare il fatto che l'ADDX modificato , testato con un margine ridotto rispetto al medesimo caso senza modifica ,presenta una frequenza di convergenza nettamente più elevata.
Per confermare la nostra ipotesi iniziale è stato fatto un confronto fra i risultati ottenuti con l'ADDX e il MADDX (Modified-ADDX, questo è il nome con cui adesso indicheremo il nostro algoritmo), prendendo a riferimento il caso 1.1 ovvero quello testato con un margine inferiori (ε = 0.15).
Sono stati considerati 100 valori di buffer valutabili in entrambi i casi,selezionati in modalità random e si è valutato lo scostamento medio:

Scostamento Assoluto	**Scostamento Relativo**
1,03%	0,7%

I risultati ottenuti sui rimanenti esempi sono riportati in tabella

Caso	2.1	2.2	2.3	3.1	3.2	3.3	4.1	4.2	4.3
Test Effettuati	1296	1296	1296	1296	1296	1296	1296	1296	1296
Non Convergenti	27	0	0	7	0	0	28	1	0
Loop	380	0	0	20	6	0	39	431	0
Convergenti	889	1296	1296	1269	1290	1296	652	1229	1296
η	0,68	1	100	0,97	0,99	1	0,5	0,95	100
ε	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
Iteraz. Max	30	30	30	30	30	30	30	30	30

E' possibile effettuare una considerazione molto importante: il nuovo criterio di convergenza rende l'ADDX praticamente operativo per la valutazione di casi asintotici in cui si impongono valori dei buffer molto elevati.
Questo fatto conferma quanto sviluppato da Burman nella formulazione dell'ADDX classico.

2.5.4 2° Miglioramento : Definizione Dominio Parametri

Il modello analitico del BAP impone che i parametri caratteristici di una linea siano soggetti a vincoli logici che permettono di far si che il modello sia il più conforme possibile alla realtà.

L'ADDX, nella sua formulazione originale, fa si che venga rispettato solamente uno dei molteplici vincoli imposti in tale modello, ovvero quello relativo alle capacità minime e massime dei buffer.

Al contrario, l'algoritmo non tiene minimamente conto del rispetto dei domini imposti sulle variabili che descrivono le macchine delle linee fittizie $L(i)$, facendo si che essi possano assumere valori qualsiasi durante le fasi di aggiornamento di up/downstream.

Analizzando accuratamente alcuni casi di overflow e loop isolati durante la sperimentazione, è stato riscontrato che buona parte di essi presenta questo tipo di evento.

Per questo motivo è stato deciso di creare un opportuno sistema di integrazione dell'algoritmo che permetta di evitare che tali parametri assumano valori non conformi per vedere se effettivamente, sia possibile ottenere un miglioramento delle prestazioni dell'ADDX.

λ Hard Costraints

I parametri $\lambda_u\ (i)$ e $\lambda_d\ (i)$ rappresentano delle approssimazioni dei tassi di rottura delle sottolinee che essi sintetizzano. Ciò implica che essi debbano assumere valori positivi o al più nulli (in quanto è possibile ipotizzare una macchina che non si guasti mai).

Il problema fondamentale è stato comprendere come comportarsi nel caso in cui tali parametro assuma valori negativi durante uno dei passaggi di aggiornamento delle differenti chiamate del MADDX. Teoricamente sarebbe infatti corretto sostituire il valore corrente negativo con $\lambda_{u,d}(i) = 0$, ma utilizzando tale valore possono venirsi a creare seri problemi a livello computazionali.

Dunque sono state individuate delle soluzioni logiche alternative di cui le più importanti sono :

- Riassegnare a tali parametri i valori da essi assunti all'iterazione precedente del MADDX.
- Riassegnare a tali parametri i valori della macchina reale a cui si riferiscono

Le due scelte possibili sono state testate per comprendere se in primo luogo queste permettano effettivamente di rendere più convergente l'algoritmo e in seguito per identificare quale fra le due sia più efficiente.
Il risultato a cui si è pervenuti è stato positivo in quanto entrambe evitano che l'algoritmo diverga a causa di valori di $\lambda_{u,d}(i)$ non accettabili. Inoltre è stato rilevato che la prima opzione rallenta fortemente il tempo di convergenza dell'algoritmo, introducendo ex novo valori di tipo "root", ovvero quelli di inizializzazione.
Nel secondo caso invece si permette all'algoritmo di non lavorare con valori negativi (e dunque di tipo "fake"), rallentandolo minimamente (si lavora con valori già in parte omogeneizzati).
Tutto ciò ha fatto optare per la seconda soluzione che è stata successivamente implementata a livello di algoritmo come di seguito: definiti λ'_u e λ'_d i valori correnti valutati per i tassi di guasto si avrà che:

$$\boldsymbol{\lambda_u}\,(\boldsymbol{i}) = \begin{cases} \lambda'_u \; se \; \lambda'_u > 0 \\ \lambda_u\,(i) \;\; se \; \lambda'_u \leq 0 \end{cases}, \qquad \boldsymbol{\lambda_d}\,(\boldsymbol{i}) = \begin{cases} \lambda'_d \; se \; \lambda'_d > 0 \\ \lambda_d\,(i) \;\; se \; \lambda'_d \leq 0 \end{cases}$$

μ Hard Costraints

Il parametro $\mu_u(i)$ rappresenta una approssimazione del tassi di riparazione $M(i)_u$ della linea di upstream a cui si riferisce.
Questo viene valutato come combinazione convessa fra il tasso di riparazione attuale μ_i e il tasso di riparazione $\mu(i-1)_u$ al quale la macchina viene riparata nel caso in cui sia rotta e il buffer $(i-1)$ risulti essere saturo.
La stessa cosa vale per il tasso μ_d, che è valutato come combinazione convessa fra $\mu_d(i+1)$ e μ_i.
Dunque valgono i seguenti vincoli :

$$MIN(\mu(i), \mu_u(i-1)) \leq \mu_u(i) \leq MAX(\mu(i), \mu_u(i-1))$$
$$MIN(\mu(i+1), \mu_d(i+1)) \leq \mu_u(i) \leq MAX(\mu(i+1), \mu_d(i+1))$$

Che sono stati implementati a livello computazionale per le linee di upstream con questo semplice sistema:

$$x = MAX(\mu(i), \mu_u(i-1))$$
$$y = MIN(\mu(i), \mu_u(i-1))$$
$$z = MIN(\mu_u(i), x)$$

Invece per le linee di downstream stato usato il seguente sistema di sostituzioni:

$$x' = MAX(\mu(i+1), \mu_d(i+1))$$
$$y' = MIN(\mu(i+1), \mu_d(i+1))$$
$$z' = MIN(\mu_d(i+1), x')$$

c Hard Costraints

$c_u(i)$ è un approssimazione della produttività $C_u(i)$ che contraddistingue le macchine della linea i-esima di upstream. Da ciò si evince che $C_u(i)$ dovrà essere inferiore rispetto alla produttività della i-esima macchina della linea $c(i)$ e allo stesso modo tale considerazione può essere estesa alle linee di downstream ricordando che $c_d(i)$ dovrà essere inferiore rispetto a $C_d(i+1)$. Dunque si hanno i seguenti vincoli :

$$0 \leq c_u(i) \leq c(i)$$
$$0 \leq c_d(i) \leq c(i+1)$$

che sono stati implementati a livello software attraverso le seguenti relazioni:

$$x = MAX(0, c_u(i))$$
$$c_u(i) = MIN(x, c(i))$$

E

$$x' = MAX(0, c_d(i))$$
$$c_d(i) = MIN(x', c(i+1))$$

Le tre modifiche proposte sono state implementate a livello di codice pilota e sono state testate sulla linea uno, che rappresenta il caso più critico del nostro set con i seguenti risultati:

Caso 1 Modifica 2		***Caso 1 Modifica***		***Caso 1.2***	
Test Effettuati	1296	Test Effettuati	1296	Test Effettuati	1296
Non Convergenti	8	Non Convergenti	11	Non Convergenti	62
Loop	298	Loop	351	Loop	802
Convergenti	990	Convergenti	934	Convergenti	402
η	0,77	η	0,72	η	0,31
ε	0.20	ε	0.20	ε	0.25
Iterazioni Max	30	Iterazioni Max	30	Iterazioni Max	30

Sebbene le prestazioni dell'algoritmo su tale esempio risultino essere sempre limitate, la modifica apportata ha incrementato l'efficacia dell'ADDX di 5 punti percentuali.
Inoltre dal confronto dei dati realizzato, la modifica non va a variare minimamente i risultati precedentemente ottenuti, ma permette di recuperare solo ed esclusivamente situazioni di overflow e loop.

2.5.5 3° Miglioramento:Cicli di Up/DownStream Molteplici

Da alcuni studi condotti da Burman sul DDX originale di Dallery, è emerso che iterando più volte le chiamate dei cicli di aggiornamento di Upstream e Downstream è possibile incrementare l'efficienza di tale algoritmo, senza perdere in precisione.
La chiamata ripetitiva di uno stesso ciclo permette di far si che l'efficacia del suo aggiornamento risulti essere nettamente più elevata, portando i singoli TP_i delle sottolinee fittizie ad omogeneizzazione completa e in un tempo globale assai più ridotto.
La causa di questo fenomeno è da ricercare nel fatto che richiamando uno stesso ciclo di aggiornamento molteplici volte è possibile restringere il dominio delle variabili in gioco, facendole oscillare in un intervallo più ridotto nell'intorno della soluzione finale.
Lo studio svolto da Burman non prevedeva l'utilizzo della tecnica delle chiamate multiple applicato all'ADDX.
Dall'analisi condotta sull'evolversi dei TP_i durante le differenti chiamate del MADDX è stato messo in evidenza come spesso l'efficacia nell'aggiornamento del ciclo di downstream risulti essere nettamente inferiore rispetto a quella di upstream, in particolare durante le primissime iterazioni.
Inoltre è stato messo in evidenza come i cicli di upstream tendono ad avere effetti di modifica più decisi sui valori delle sottolinee più lontane dalla macchina reale di monte, mentre al contrario quelli di downstream tendono a modificare con più efficacia le prime sottolinee equivalenti.
Questo causa la presenza di Loop e Overflow, come mostrato precedentemente.

Tutto ciò ha indotto a estendere la tecnica appena descritta al MADDX, non tanto per incrementare ulteriormente la velocità di convergenza dell'ADDX, bensì per aumentarne la frequenza di convergenza.
Per farlo sono state considerate le linee precedentemente testate ed è stato modificato il software pilota in modo tale da impostare una chiamata multipla dei cicli di upstream e downstream per un numero di 5 iterazioni.

Algoritmo risolutivo:

1. *Inizializza le variabili*
2. *Ripeti Q volte l'aggiornamento delle variabili di upstream.*
3. *Ripeti Q volte l'aggiornamento delle variabili di downstream.*
4. *Ripeti i passi 3 e 4 fino a quando non viene soddisfatta la condizione di stop.*

Il risultato finale è stato molto soddisfacente per due motivi principali:

- Incremento sensibile della frequenza di convergenza.
- Allineamento dei risultati con i valori precedentemente trovati.

L'unico problema riscontrato con la modifica attuata è legato al tempo globale necessario all'algoritmo per convergere.
Non è stato possibile identificare un sistema con cui poter definire il numero ottimale di iterazioni multiple da eseguire e dunque durante la sperimentazione sono stati utilizzati valori di Q di sicurezza con cui assicurare che l'aggiornamento potesse essere efficace.
Tutto ciò ha rallentato fortemente l'esecuzione del MADDX.
L'aspetto positivo è che nel momento in cui l'aggiornamento legato ad un singolo ciclo è effettivamente terminato, l'aggiornamento dovuto alle iterazioni successive diventa nullo e dunque non vi è il rischio di poter falsare il lavoro svolto dall'ADDX durante la valutazione, sebbene come detto precedentemente sia possibile che possa manifestarsi uno spreco evidente a livello di tempo.

2.5.6 Riepilogo Prestazioni MADDX

Dopo aver apportato le modifiche precedentemente descritte all'ADDX, esse sono state testate sugli esempi di riferimento precedentemente descritti con i seguenti risultati:

Caso 1 ADDX mod.		***Caso 1.2***		***Caso 1 MADDX***		***Caso 1 MADDX***	
Casi Testati	1296	Casi Testati	1296	Casi testati	4096	Casi testati	4096
Non Convergenti	11	Non Convergenti	62	Non Convergenti	0	Non Convergenti	0
Loop	351	Loop	802	Loop	45	Loop	0
Convergenti	934	Convergenti	402	Convergenti	4041	Convergenti	4096
η	0,72	η	0,31	η	0.99	η	1
ε	0.20	ε	0.25	ε	0.15	ε	0.25
Iterazioni Max	30	Iterazioni Max	30	Iterazioni Max	30	Iterazioni Max	30

La tabella riassume l'evolversi delle prestazioni dei vari algoritmi descritti sul caso 1, considerando valori differenti di ε e mantenendo costante il numero di iterazioni massime dell'algoritmo.

E' significativo come il MADDX, mandato in esecuzione con valori più stringenti di ε e su un numero più elevato di casi sia sempre più prestante rispetto al primo ADDX.

I risultati ottenuti sono stati analizzati per comprendere se, la maggiore convergenza del MADDX potesse essere generata da una minore precisione di valori ottenuti.

Dall'analisi spot eseguita confrontando valori valutati con il MADDX e valori ottenuti per mezzo della simulazione si è giunti ai seguenti risultati:

CASO	ERR.Medio	ERR.Max
ADDX mod.-MADDX(0.15)	0,83%	4,2%
MADDX(0.15)-MADDX(0.25)	3,76%	6,2%

I seguenti risultati permettono di validare il modello e di accettarlo come sistema ufficiale di valutazione di linee produttive asincrone per la risoluzione del BAP.

Per completezza vengono mostrati i risultati ottenuti mettendo a confronto l'ADDX con il MADDX per i rimanenti esempi utilizzati durante la sperimentazione:

Caso ***MADDX***	2.1	2.2	2.3	3.1	3.2	3.3	4.1	4.2	4.3
Test Effettuati	1296	1296	1296	1296	1296	1296	1296	1296	1296
Non Convergenti	2	0	0	1	0	0	9	1	0
Loop	76	0	0	6	6	0	41	8	0
Convergenti	1218	1296	1296	1289	1290	1296	1244	1287	1296
η	0,94	1	100	0,99	0,99	1	0,96	0,99	100
ε	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
Iteraz. Max	30	30	30	30	30	30	30	30	30

.

2.6 Metodo di disaggregazione

Le tecniche di aggregazione e di decomposizione non rappresentano gli unici due metodi di valutazione di linee asincrone fino ad ora sviluppati.

La ricerca analitica si sta muovendo su una strada alternativa che si basa sull'idea di "trasformare" linee asincrone in linee equivalenti di tipo sincrono.

La ragione per cui si sta tentando di seguire questa via è piuttosto semplice, ovvero la possibilità di sfruttare gli algoritmi di decomposizione per linee omogenee che risultano essere assai più efficienti e precisi, rispetto a quelli realizzati per linee asincrone.

Le tecniche in questione rientrano nella macrofamiglia definita di "Disaggregazione".

L'idea alla base di tale tecnica è quella per cui data una linea di tipo non omogeneo, ogni sua macchina possa essere rimpiazzata da due differenti macchine separate da un buffer di capacità nulla contraddistinte dallo stesso tempo ciclo τ.

In questo modo ,sebbene la linea equivalente risulterà essere di dimensioni raddoppiate rispetto all'originale,sarà composta a sua volta da macchine contraddistinte tutte dallo stesso tempo ciclo.

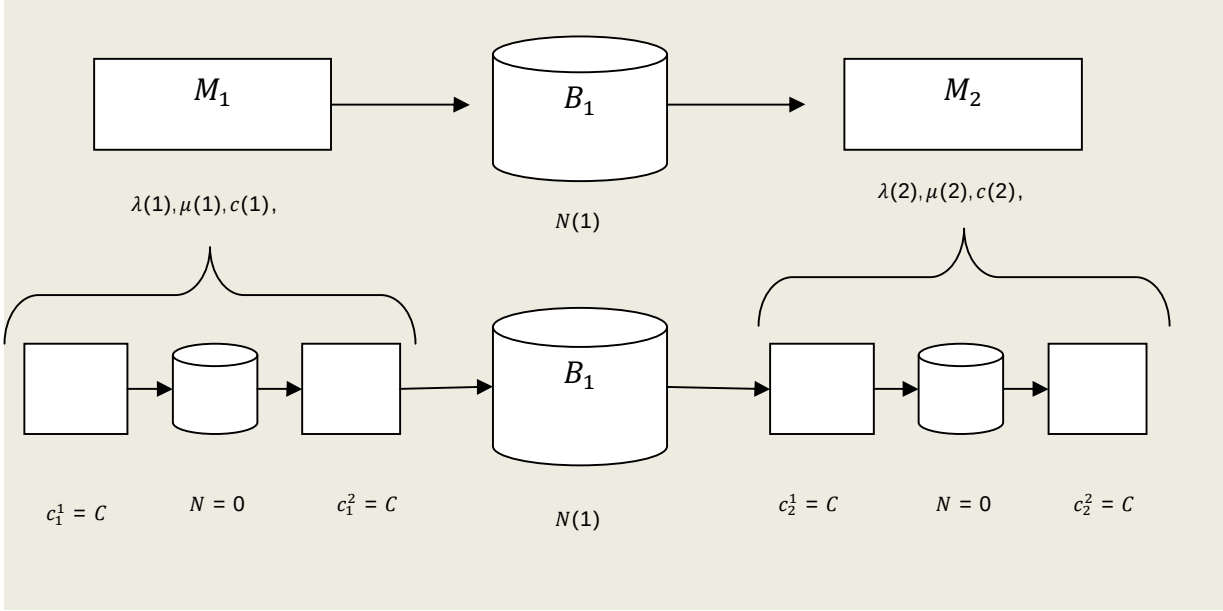


Figura 6, Metodo di Disaggregazione

Una delle due macchine ingloberà in se le caratteristiche di "unreliability" della macchina originale, mentre la seconda rappresenterà la sua produttività in assenza di rotture.

Queste due macchine disporranno della capacità produttiva della macchina più veloce della linea (dunque quella per cui τ = min ($\tau_1, \ldots, \tau_n$)).
Dunque una volta definiti i tassi di guasto/riparazione delle due macchine equivalenti come:$\lambda_i^1, \lambda_i^2, \mu_i^1, \mu_i^2$ essi saranno così valutati:

$$\mu_i^2 = \mu_i$$

$$\lambda_i^2 = \frac{\tau_i}{\tau}\lambda_i$$

$$\frac{\mu_i^1}{\lambda_i^1 + \mu_i^1} = \frac{\tau}{\tau_i}$$

$$\lambda_i^1, \mu_i^1 \gg \lambda_i^2, \mu_i^2$$

La quarta condizione deriva dal fatto che la prima macchina rappresenta il tempo ciclo della macchina reale,mentre la seconda ingloba in se il comportamento di indisponibilità della macchina originale.
I parametri della prima macchina devono assumere valori molto più elevati rispetto a quelli della seconda.
Più si riesce e replicare tale situazione e più l'accuratezza della disaggregazione sarà elevata. Le analisi condotte con la simulazione hanno fornito risultati molto simili a quelli valutati con tale sistema, con un errore medio del 6% e picchi massimi del 10%, che sebbene siano contenuti, ancora non si avvicinano a quelli ottenibili con le tecniche di decomposizione precedentemente descritte.

2.7 Metodo di omogeneizzazione

Una tecnica alternativa alla disaggregazione,più semplice e performante è detta di omogeneizzazione .Tale trasformazione rimpiazza ogni macchina della linea originale con una macchina equivalente, in modo tale che al termine delle sostituzioni si avrà una linea composta da macchine tutte contraddistinte dallo stesso tempo ciclo τ, dunque una linea di tipo omogeneo trattabile con le classiche tecniche di decomposizione.
Detti λ_i^e, μ_i^e i parametri della i-esima macchina equivalente,questi parametri devono essere scelti in modo tale che il comportamento della linea equivalente sia il più simile a quello della linea originale.

Dunque tali parametri devono essere valutati in modo adeguato seguendo le seguenti sostituzioni:

$$\mu_i^e = \mu_\mathrm{i}$$

$$\lambda_i^e = \lambda_i + \frac{\tau_i - \tau}{\tau}(\lambda_i + \mu_i)$$

Come si può vedere il tasso di riparazione di ogni singola macchina viene mantenuto costante rispetto al valore che questo assumeva sulla macchina di origine, mentre ciò che viene adattato è il tasso di rottura, che verrà incrementato/decrementato in funzione del fatto che il $\tau_i > \tau$ o viceversa .
In parole povere, macchine più lente vengono penalizzate con un tasso di rottura più elevato (a cui logicamente corrisponde un MTBF più limitato),che comporterà un'efficienza interna ridotta e conseguentemente un TP isolato più limitato.
L'omogeneizzazione è una trasformazione assai più grossolana rispetto alla disaggregazione e sebbene permetta di ottenere risultati più velocemente e con una difficoltà computazionale estremamente ridotta ,tale tecnica pecca di precisione .

3 Ottimizzazione Performance Linee Asincrone

Nei capitoli precedenti abbiamo descritto le entità principali che compongono una linea produttiva, sono state identificate le principali classificazioni delle linee stesse e sono stati introdotti i coefficienti di prestazione con cui è possibile valutarle.
In ultimo luogo è stato fatto uno screening delle varie tecniche con cui tali prestazioni possono essere effettivamente valutate e partendo dall'ADDX di Burman si è giunti alla definizione di un nuovo strumento definito MADDX, sempre appartenente alla famiglia delle tecniche di decomposizione, ma molto più performante e valido.
Tale modello permette di calcolare analiticamente il Throughput di linee asincrone, al variare delle capacità massima dei buffer presenti fra le macchine che le compongono.
L'idea alla base di questa sezione dell' elaborato è quello di sfruttare il modello precedentemente analizzato ed affiancarlo ad un opportuno sistema di ottimizzazione in modo tale da riuscire a definire la composizione ideale dei buffer di una linea.
Questo permetterà di incrementare le prestazioni di questo tipo di sistema produttivo solamente agendo sul suo layout e non sulle macchine che lo compongono (evitando ad esempio la sostituzione delle stesse con macchine più efficienti).
Per raggiungere l'obiettivo che ci siamo prefissi di raggiungere dovremo seguire i seguenti step:

- Definire analiticamente il problema da risolvere;
- Definire opportuni sistemi di ottimizzazione;
- Applicare tali sistemi al problema definito.

3.1 Definizione del Problema (Buffer Allocation Problem)

L'ottimizzazione delle Performance di una linea produttiva è un problema descrivibile attraverso il seguente modello :

$$\begin{cases} Max\ Throghput\ f(\lambda_1, \dots, \lambda_k, \mu_1, \dots, \mu_{k,} c_1, \dots c_k, BC_1, \dots, BC_{k-1}) \\ \lambda_1, \dots, \lambda_k \geq 0 & (a) \\ \mu_1, \dots, \mu_{k,} > 0 & (b) \\ c_1, \dots c_k > 0 & (c) \\ c_i \neq c_{i+1}\ \forall\ i: [1 \dots k-1] & (d) \\ \sum_{i=1}^{k-1} BC_i \leq B_{tot} & (e) \\ BC_i \epsilon N, BC_i \geq 0 & (f) \end{cases}$$

Esso consiste in un problema di massimizzazione vincolata in cui:

- La funzione obiettivo, ovvero il Throughput della linea, è funzione dei parametri caratteristici delle macchine che la compongono ovvero il tasso di guasto(λ), il tasso di riparazione (μ) e la capacità produttiva (c) ed è funzione delle capacità massime dei buffer presenti fra di esse. Tale funzione è valutata tramite il modello analitico (MADDX) trattato nel capitolo precedente.
- Il vincolo (a) ci dice che i tassi di guasto delle varie macchine possono assumere valori positivi o al più nulli, includendo dunque le situazioni per cui una macchina non sia soggetta mai a guasto ($\lambda = \infty$), o sia praticamente sempre rotta ($\lambda = 0$).
- Il vincolo (b) ci dice che i tassi di riparazione delle varie macchine devono essere obbligatoriamente positivi, questo implica che non sia possibile che si manifesti una situazione per cui una macchina necessiti di una riparazione dalla durata infinita (in tal caso la linea non entrerebbe più in funzione).
- Il vincolo (c) rappresenta una condizione logica da soddisfare sempre, infatti disporre di una macchina con capacità produttiva nulla vorrebbe dire non avere la macchina e non fare funzionare la linea. Logicamente avrebbe ancor meno senso avere una capacità produttiva negativa.
- Il vincolo (d) è fondamentale perché ci permette di analizzare solo linee produttive di tipo asincrono, ovvero quelle linee per cui due macchine successive non possiedono la stessa capacità produttiva (o per vedere la stessa situazione sotto un altro punto di vista, macchine caratterizzate da

due tempi ciclo differenti). Il vincolo non impone che tutte le macchine abbiano tempi ciclo differenti ma solo due successive. Questo permetterà di valutare la funzione obiettivo utilizzando il MADDX.

- Il vincolo (e) può essere interpretato come vincolo di spazio: uno stabilimento reale non dispone di spazio infinito destinato all'allocazione di semilavorati, bensì di spazi limitati.
- Il vincolo (f) è un vincolo logico, infatti i valori assunti dai buffer dovranno essere interi positivi, in quanto trattando processi per parti non potremo accettare di avere a stock componenti "parziali" o stock negativi. Tale vincolo dovrebbe invece essere eliminato nel caso in cui si analizzasse un processo in continua come la produzione della carta e della birra.
- Sulle capacità dei buffer è stato posto un vincolo di capacità massima, infatti avendo imposto che il buffer globale non possa superare una determinata soglia, si assume implicitamente che nel caso in cui tutta questa quantità venga allocata su un'unica posizione, tali capacità andranno a coincidere. Tale vincolo non è espresso esplicitamente perché deriva in toto dal vincolo (e).

Analisi delle soluzioni

L'insieme delle possibili soluzioni del problema appena descritto rappresenta una frontiera $(k-1)$ dimensionale, con k numero di macchine che compongono la linea.

Fino a pochi anni fa era diffusa la convinzione per cui questo spazio fosse convesso sebbene tale teoria non fosse stata validata da opportune analisi numeriche. Tale supposizione è stata smentita dagli studi condotti da Dallery.

Il fatto di non poter lavorare su un dominio continuo, bensì discreto di variabili fa si che non sia possibile utilizzare i classici metodi di ottimizzazione legati alla ricerca operativa (algoritmo del gradiente,Newton ecc.ecc.) per la risoluzione del problema, ma al contrario è più facile fare affidamento ad altre classi di algoritmi che andremo a descrivere successivamente.

Algoritmi di Ottimizzazione

Una volta definito il problema da risolvere e avendo scelto un opportuno sistema di valutazione della funzione obiettivo è necessario definire un sistema con cui ottimizzare tale funzione, nel rispetto dei vincoli imposti.
Le considerazioni espresse sul tipo di spazio di soluzioni da esplorare è scaturita la decisione di fare affidamento a metodi di ricerca locale di tipo metaeuristico, poiché tali sistemi rappresentano in assoluto gli approcci di maggior successo nell'ottimizzazione di problemi combinatori.
In generale un metodo euristico è un approccio alla risoluzione di un problema che non segue un chiaro percorso di ricerca (come nei metodi algoritmici), ma che piuttosto permette di arrivare all'obiettivo affidandosi all'intuito, replicando e sfruttando lo stato temporaneo delle circostanze al fine di generare nuova conoscenza.
Un metodo meta euristico permette di risolvere un' ampia gamma di problemi computazionali combinando differenti procedure a loro volta euristiche ed ottenendo a sua volta procedure più robuste ed efficienti.
Ricerca locale è il nome che viene dato comunemente al gruppo di metodi che iterativamente ripetono il rimpiazzamento di una soluzione corrente s con una soluzione nuova s^* fino a quando non viene soddisfatta una opportuna condizione di stop, per mezzo di cui si massimizza/minimizza una determinata funzione obiettivo.
Potremmo spiegare il funzionamento generale di tali sistemi per mezzo dell'esempio del cuoco maldestro: volendo creare un piatto nuovo, che stuzzichi l'appetito dei propri ospiti, il cuoco di poca esperienza scrive una nuova ricetta e la propone una prima volta misurando l'apprezzamento ai tavoli. In seguito varierà la dose di un ingrediente per verificare se il piatto può essere migliorato e procede così fino al momento in cui arriva alla formulazione ottima della ricetta: procedendo iterativamente riuscirà nel tempo, per mezzo di opportuni tentativi a realizzare un piatto ottimo, basandosi dunque sui feedback ottenuti dalla sala e degli effetti delle modifiche passate sul piatto.

1. **Hill-Climbing**: E' il più semplice algoritmo di ricerca locale, sviluppato nel 1960 e usato nei problemi di schedulazione. Una soluzione viene accettata in $N(s)$, se la sua funzione di fitness è migliore rispetto alla corrente in $N(s)$.

Non richiede la definizione di alcun parametro di controllo, converge velocemente, ma spesso le soluzioni definite sono di scarsa qualità, perché tende bloccarsi su ottimi locali.

2. **Simulated Annealing**:è un estensione e un miglioramento dell'Hill Climbing, che permette di ottenere soluzioni di qualità superiore. Il funzionamento è speculare, ma si differenzia per il fatto che soluzioni non ottimali possono essere incluse in $N(s)$,con una probabilità funzione di $\delta = f(s) - f(s^*)$.Questo sistema permette all'algoritmo di "svincolarsi" da ottimi locali e dunque di non arenarsi.
3. **Threshold Acceptance Algorithm**: è una variante del Simulated Annealing in cui la probabilità di accettazione di una soluzione non ottimale non è funzione della soluzione corrente, bensì dipendente da un parametro δ stabilito a priori.

L'obiettivo di questa sezione dell'elaborato sarà quello di testare e mettere a confronto tramite un'analisi di Benchmarking due algoritmi di risoluzione di ultima generazione: da questa attività emergerà lo strumento ottimizzatore che utilizzeremo ufficialmente nella risoluzione del BAP.

3.2 Degraded Ceiling

Il DC basa la propria ricerca sul principio del "Replacing", il cui funzionamento in un contesto di massimizzazione è sintetizzato in figura 7: dopo aver inizializzato un parametro di ricerca ΔL, si generi una qualsiasi soluzione ammissibile S_{start} del problema e se ne valuti la funzione obiettivo $f(S_{start})$. Con tale valore si inizializza la soglia (inferiore) L. La soluzione S_{start} entrerà a far parte di una memoria $N(s)$ definita "Neighborhood"(vicinato) contenente le migliori soluzioni trovate. A questo punto una nuova soluzione s^*,ottenuta partendo da quelle presenti in $N(s)$, se soddisferà determinate condizioni entrerà in $N(s)$ sostituendo la soluzione s in essa compresa. In caso contrario si valuterà una nuova soluzione s^*.

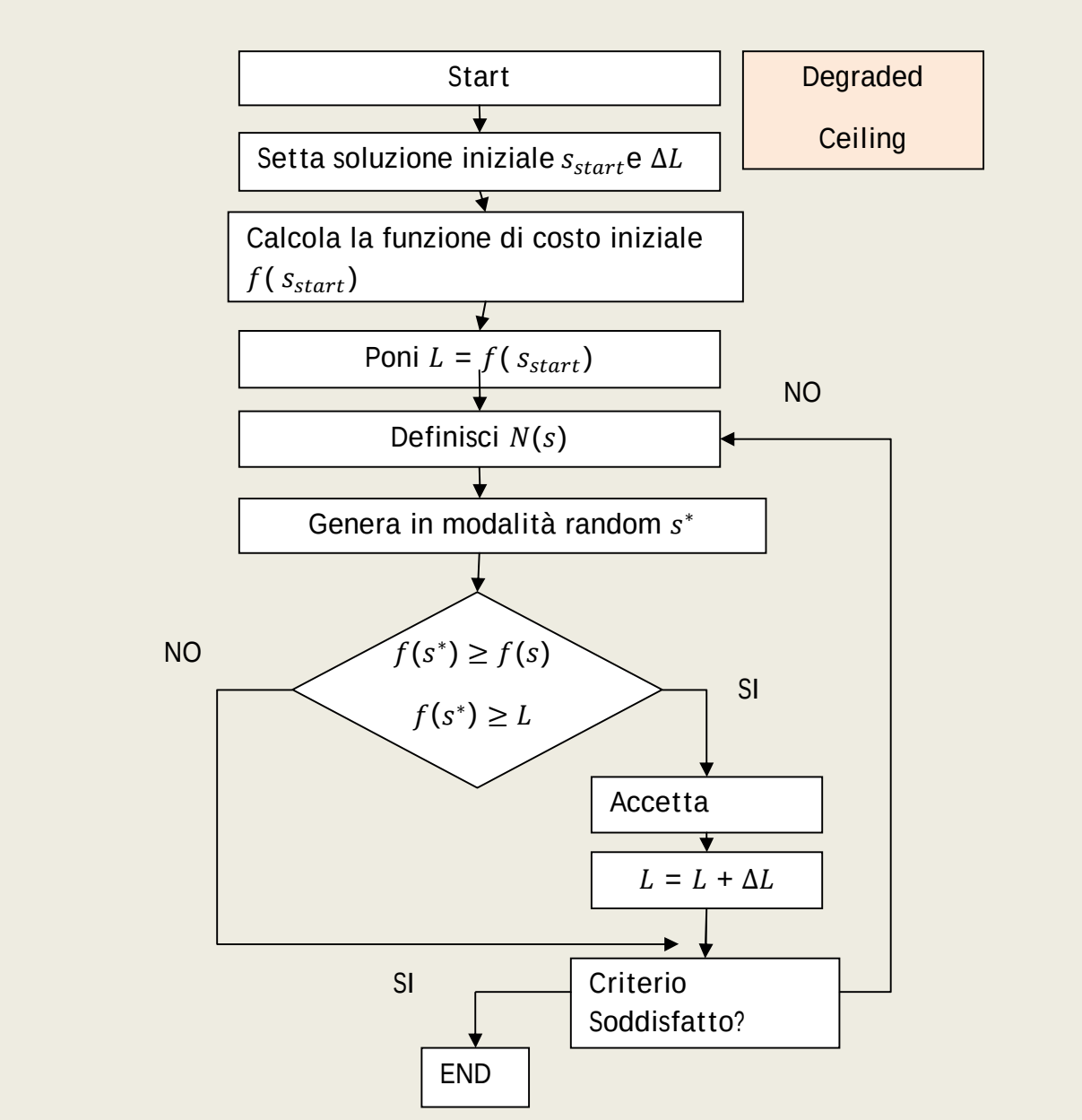


Figura 7, Flow Chart Degraded Ceiling

Le condizioni su cui si basa il "Replacing" sono due: o s^* ha una funzione obiettivo superiore alla soglia inferiore L o la sua funzione obiettiva è quantomeno migliore delle soluzioni contenute nella memoria $N(s)$: $f(s^*) \geq f(s)$ con s $\in N(s)$ (in questo caso la soluzione trovata rappresenta una strada di ricerca più promettente).
Ogni volta che una nuova soluzione entra in $N(s)$, L viene incrementato di ΔL: in questo modo si rende il campo di ricerca più ristretto e si permette all'algoritmo di convergere.
La ricerca termina quando viene soddisfatta una determinata condizione.
La particolarità del DC consiste nel fatto di utilizzare un Neighborhood composto da una sola soluzione: se da un lato questa scelta limita l'efficacia di ricerca dell'algoritmo, dall'altro gli permette di identificare con estrema costanza soluzioni sub-ottimale e di essere estremamente efficiente.
Lo schema proposto è del tutto generale e adattabile a molteplici casi di studio per mezzo di semplici modifiche: quella necessaria è logicamente quella legata alla definizione di un opportuno sistema di generazione random di nuove soluzioni s^* in modo tale da adeguare lo strumento al tipo di soluzione/problema affrontato.
E' molto importante comprendere il motivo per cui esiste una duplice condizione di accettazione di nuove possibile soluzioni in N(s):

- La condizione $f(s^*) \leq L$ ci dice che se una soluzione corrente presenta una fitness function migliore rispetto alla soglia prestabilita essa è candidata ad essere l'ottimo o almeno rappresenta una buona base di esplorazione e dunque dovrà entrare a fare parte di N(s).
- La condizione $f(s^*) \leq f(s)$ permette invece di "trovare la strada giusta", nel momento in cui l'algoritmo "perde la bussola", ovvero permette di abbandonare una ipotetica via di peggioramento delle prestazioni (la soluzione (s)) e riproporre una nuova ricerca su (s^* la cui funzione di fitness risulta essere migliore. Dunque può essere letta come una condizione di re-indirizzamento dell'algoritmo.

In realtà uno dei problemi fondamentali del DC risiede proprio in questo sistema di aggiornamento, perché dato che il problema da noi analizzato non è di tipo convesso, è possibile che il DC possa arenarsi in soluzioni ottime locali.

3.2.1 Degraded Ceiling applicato al BAP

Come detto precedentemente, per applicare il DC al BAP sarà necessario adattarlo in maniera opportuna, identificandone le modifiche necessarie e implementandole a livello di sistema.

N(s):

Il modello di Neighborhood scelto è lo stesso identificato nel modello originale, composto da un'unica soluzione

Inizializzazione N(s):

La soluzione iniziale (s) verrà definita distribuendo in maniera uniforme la capacità totale B_{tot} fra le varie posizioni del vettore BC: $(BC_1, \dots, BC_{k-1})$ distribuendo le rimanenze sulle posizioni centrali.

Tale decisione viene presa per permettere all'algoritmo di partire in una condizione il più generale possibile, in modo tale che possa disporre del numero maggiore di vie di ricerca possibile e che non gli vengano precluse a priori intere gamme di soluzione.

Se noi paragonassimo lo spazio delle soluzioni ad un campo definito da una linea chiusa che debba essere perlustrato, non faremmo altro che far partire il DC (il nostro esploratore) dal "baricentro" di tale area come mostrato in Figura 8.

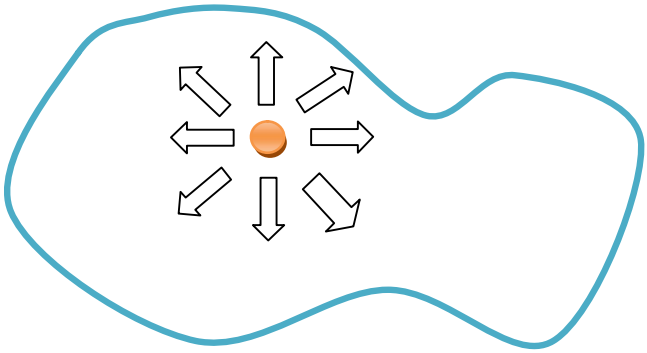


Figura 8, Punto di partenza DC

Aggiornamento di N(s):

Le nuove soluzioni vengono generate nel seguente modo:data una soluzione del tipo s: $(B_1, \dots, B_{k-1})$, inclusa in $N(s)$ si scelgono due posizioni p e q con $p \neq q$. Le corrispondenti allocazioni (B_p, B_q) verranno modificate nel seguente modo:

$$(B_p = B_p - 1), (B_q = B_q + 1).$$

In questo modo viene rispettato il vincolo di B_{tot} e si aggiorna (s) in maniera pratica e veloce.

Il sistema di aggiornamento ci mostra come a piccoli passi il DC riesca sempre ad esplorare in maniera approfondita lo spazio delle soluzioni, sebbene aggiornando due sole variabili del vettore buffer, questo possa soffrire di lentezza e pericolo di rimanere impantanato su ottimi locali.

Aggiornamento di L

L'aggiornamento della soglia limite viene realizzato ogni qualvolta venga immessa una nuova soluzione in N(s), in modo tale che quando il DC intraprende una nuova strada di ricerca, si limiti il campo della stessa. In questo modo si evita di ricercare soluzioni su strade poco promettenti e si rende più efficiente l'algoritmo.

3.2.2 Validazione del Buffer Degraded Ceiling

Una volta descritto a livello teorico il Buffer Degraded Ceiling si è passati alla sua implementazione e alla successiva fase di test.

La validazione è stata realizzata per mezzo di un software applicativo, appositamente sviluppato in linguaggio Delphi: tale strumento sfrutta il modulo di valutazione di linee asincrone precedentemente realizzato per la validazione del MADDX.

L'obiettivo che si è voluto raggiungere è stato quello di comprendere come variano le prestazioni del DC al variare dei parametri fondamentali di computazione ovvero ΔL e I, numero di iterazioni globali dell'algoritmo.

Questo analisi ha permesso inoltre di definire il setting ottimale di tali parametri con cui ottenere le migliori soluzioni durante la ricerca.

Il DC è un algoritmo di ottimizzazione di tipo parametrico: quando viene mandato in esecuzione è necessario settarne il fattore ΔL ,che limita in maniera più o meno elevata l'accesso a nuove possibili soluzioni all'interno di N(s).

L'obiettivo finale della nostra analisi sarà quello di stabilire il setting ottimale del DC, che verrà utilizzato durante l'attività di Benchmarking con l'Harmony Search.

L'insieme dei nostri esperimenti è stato così organizzato:

1. Selezione di casi di studio
2. Analisi dei dati e sintesi

Selezione dei casi

E' stato deciso di testare l'algoritmo su linee di tre differenti dimensioni, composte rispettivamente da 5, 10 e 15 macchine e per ogni caso selezionato appartenente alle singole categorie eseguire esperimenti su tre differenti valori di ΔL.
Questo ci permetterà di comprendere eseguendo delle semplici statistiche sui dati rilevati, quale sia l'influenza di un settaggio più o meno fine sui valori finali ottenuti.
Il criterio con cui sono stati generati i valori dei parametri delle macchine che compongono le linee scelte è piuttosto semplice: per ogni categoria sono stati generati tre differenti esempi, il primo dei quali presenta macchine dall'elevata efficienza e piuttosto omogenee tra di loro, il secondo con macchine omogenee ma dall'efficienza ridotta, più un terzo esempio generato in maniera random.
I casi testati sono riportati in allegato, al capitolo 7.2.

Analisi Dati e sintesi

La valutazione delle prestazioni dell'algoritmo è stata effettuata tenendo conto dei seguenti fattori:

- **Valore Max:**questo fattore ci permette di comprendere la capacità dell'algoritmo di assicurare l'ottenimento dell'ottimo mandandolo in esecuzione per un numero determinato di volte.
- **Valore Min:** ci permette di comprendere quale sia la precisione dell'algoritmo.
- **Valore Medio:** ci permette di comprendere l'affidabilità media dell'algoritmo.
- **Cicli di Convergenza:**è un parametro che ci permette di valutare l'efficienza dell'algoritmo.

Di seguito proponiamo i valori ottenuti dai casi studiati, brevemente commentati.

I primi due casi analizzati rientrano nella categoria M5 (linee formate da 5 macchine), composte da macchine efficienti e dalle produttività molto simili l'una con l'altra. Inoltre nei seguenti casi è stato deciso di allocare un buffer totale limitato (20 elementi).

Caso	Delta	Numero Prove	V.Max	V.Min	V.Medio	Cicli Conv
Linea 1	0,0005	100	5,287	5,287	5,287	70
Linea 1	0,00005	100	5,287	5,287	5,287	300
Linea 1	0,000005	100	5,287	5,287	5,287	1600
Linea 2	0,0005	100	3,7819	3,7819	3,7819	80
Linea 2	0,00005	100	3,7819	3,7819	3,7819	350
Linea 2	0,000005	100	3,7819	3,7819	3,7819	1700
Linea 3.1	0,0005	100	3,489	3,489	3,489	200
Linea 3.1	0,00005	100	3,489	3,489	3,489	1600
Linea 3.1	0,000005	100	3,489	3,481	3,486561	3000

Come è possibile osservare dalla tabella, in questa situazione, il DC converge molto velocemente, fornendo nei tre differenti casi testati risultati uniformi.
Utilizzando ΔL ridotti, l'algoritmo necessita di un numero di cicli maggiore per convergere, in quanto imponendo delle condizioni meno stringenti questo esplora un numero di strade più elevato prima di convergere.
Questa è la ragione per cui nel caso 3.1 testato con valore minimo di ΔL, l'algoritmo presenta dei segni di cedimento.
Questi esempi dimostrano che nei casi di difficoltà computazionale ridotta il DC risulta essere molto prestante, ma non appena si aumenti lo spazio di ricerca, è evidente il fatto che questo necessiti di maggiore inerzia per raggiungere l'obiettivo sperato.
Il caso 3.1, propone una linea leggermente più complessa, le cui macchine sono state generate in maniera random.
Anche in questo caso si è riscontrato il comportamento tipico descritto precedentemente, tranne nel caso di ΔL = 0,000005, in cui l'algoritmo ha avuto dei problemi computazionali, legati sostanzialmente al fatto che avendo ridotto nettamente il ΔL, avremmo dovuto incrementare in maniera superiore il numero di cicli di esplorazione.

Per comprendere se tale problema potesse essere evitato la linea è stata testata ulteriormente ad un numero di cicli maggiore con i seguenti risultati a dimostrazione del fatto che aumentando il numero di cicli di valutazione è possibile incrementare fortemente le prestazioni del DC nella ricerca dell'ottimo.:

Caso	Delta	Numero Prove	V.Max	V.Min	V.Medio	Cicli Conv
Linea 3.1	0,000005	100	3,489	3,483	3,4873	8000

La stessa linea è stata utilizzata per comprendere come il DC potesse comportarsi aumentando il buffer globale da suddividere fra le varie posizioni:

Caso	Delta	Numero Prove	V.Max	V.Min	V.Medio	Cicli Conv
Linea 3.2	0,0005	100	3,526	3,526	3,526	140
Linea 3.2	0,00005	100	3,526	3,526	3,526	1600
Linea 3.2	0,000005	100	3,527	3,505	3,514	5000

Sebbene anche in questo caso si sia riscontrata la debolezza intrinseca di una ricerca più fine, l'esempio ha messo in luce una verità molto importante ovvero la possibilità di trovare attraverso una ricerca di "Fine Tuning" soluzioni globali migliori rispetto al caso in cui si effettuino ricerche più grezze.
Se infatti l'ottimo globale nei primi due casi si attestava sul valore TP = 3,526, imponendo un ΔL di un ordine inferiore di grandezza si è arrivati all'ottenimento di TP = 3,527.
Comprendendo come il numero di cicli totali influisca fortemente sull'ottenimento di buone soluzioni finali, è stato deciso di effettuare le successive prove settando il numero globale di iterazioni su valori nettamente superiori, cercando sempre di non oltrepassare le soglie tipiche di convergenza dell'algoritmo per ogni singolo caso.
Per riuscire nello scopo sono state create delle prove pilota che hanno permesso di definire il numero medio di cicli di convergenza, su cui impostare vere e proprie campagne di ricerca.
Tali prove sono state realizzate attraverso l'utilizzo di un software pilota e sono consistite nell'esecuzione dell'algoritmo per un numero di volte limitato (settato su 5 tentativi)che hanno messo in evidenza il punto medio di convergenza del DC nei singoli casi.

In seguito non è stato fatto altro che settare le vere prove, su un valore di cicli prudenziale maggiore di 1000 unità rispetto al massimo dei valori identificati precedentemente.
Questo ha permesso di realizzare prove intrinsecamente valide, ma mai inefficienti
Gli ultimi esperimenti condotti su linee del tipo M5 hanno continuato a dare risultati conformi alle attese:

Caso	Delta	Numero Prove	V.Max	V.Min	V.Medio	Cicli Conv
Linea 4	0,0005	100	3,500	3,442	3,483	5000
Linea 4	0,00005	100	3,500	3,492	3,495	8000
Linea 4	0,000005	100	3,500	3,493	3,497	12000
Linea 5	0,0005	100	4,661	4,622	4,654	5000
Linea 5	0,00005	100	4,703	4,643	4,676	8000
Linea 5	0,000005	100	4,703	4,660	4,677	12000

I successivi casi studiati sono relativi a linee di tipo M10: come è ben visibile, a parità di ΔL è stato deciso di utilizzare valori di I più elevati per rendere l'algoritmo più efficace, a discapito logicamente della sua efficienza intrinseca.
E' da sottolineare l'importanza dell'esempio relativo alla Linea 8, con valore di $\Delta L = 9 * 10^{-5}$, in cui si può vedere un comportamento "anomalo" in quanto con un parametro di ricerca più grezzo è stato raggiunto un ottimo globale migliore rispetto ad un ΔL ridotto.
In realtà nel caso successivo molto probabilmente non sono stati raggiunti ottimi locali perché il numero globale di iterazioni necessarie ad una corretta convergenza era più elevato.
E' inoltre fondamentale aggiungere che nel momento in cui la complessità intrinseca delle linee studiate aumenta, il DC non riesce più a mantenere la capacità di fornire risultati "buoni", ovvero aumenta lo spread fra valori massimi e minimi e si riducono gradualmente i valori medi ottenibili.

Caso	Delta	Numero Prove	V.Max	V.Min	V.Medio	Cicli Conv
Linea 6	0,0005	100	5,200	5,167	5,170	5000
Linea 6	0,0005	100	5,200	5,167	5,170	5000
Linea 6	0,00005	100	5,208	5,167	5,170	8000
Linea 6	0,000005	100	5,210	5,143	5,162	12000
Linea 7	0,0005	100	2,423	2,051	2,262	5000
Linea 7	0,00009	100	2,438	2,134	2,29	8000
Linea 7	0,00005	100	2,687	2,453	2,429	12000
Linea 8	0,0005	50	1,422	0,773	1,001	5000
Linea 8	0,00009	50	3,170	1,009	1,669	8000
Linea 8	0,00005	50	3,056	0,991	1,661	12000

Rivolgendoci agli esempi condotti su linee del tipo M15 vediamo come il DC continui ad essere molto performante nei casi di studio più standard e perda di efficacia all'aumentare della complessità dei casi di studio.

Caso	Delta	Numero Prove	V.Max	V.Min	V.Medio	Cicli Conv
Linea 9	0,0005	50	5,057	5,07	4,858	12000
Linea 9	0,00009	50	5,112	5,032	5,081	12000
Linea 9	0,00005	50	5,121	5,035	5,085	12000
Linea 10	0,0005	50	5,712	4,937	5,341	12000
Linea 10	0,00009	50	5,723	5,189	5,523	12000
Linea 10	0,00005	50	5,913	5,357	5,5670	12000
Linea 11	0,0005	50	0,603	0,419	0,5275	12000
Linea 11	0,00009	50	0,619	0,422	0,585	12000
Linea 11	0,00005	50	0,654	0,438	0,5275	12000

3.2.3 Riassunto caratteristiche del Buffer Degraded Ceiling.

1. La velocità di convergenza dell'algoritmo è funzione del ΔL di ricerca stabilito: minori valori di ΔL corrispondono ad un numero maggiore di iterazioni necessarie per raggiungere la convergenza.
2. Nel momento in cui l'algoritmo arriva alla convergenza ogni iterazione ulteriore diviene superflua.
3. ΔL di ricerca ridotti permettono di ottenere soluzioni di picco migliori rispetto a ΔL più elevati. La causa di questo comportamento è da ricollegare al fatto che nel momento in cui la soluzione limite L cresce lentamente, si da la possibilità ad un numero molto elevato di soluzioni di entrare in $N(s)$, rendendo la ricerca più estesa. Allo stesso tempo però, sarà necessario "dare più tempo" all'algoritmo per trovare la giusta strada di convergenza. Il comportamento tipico definito è sintetizzato nel seguente grafico:

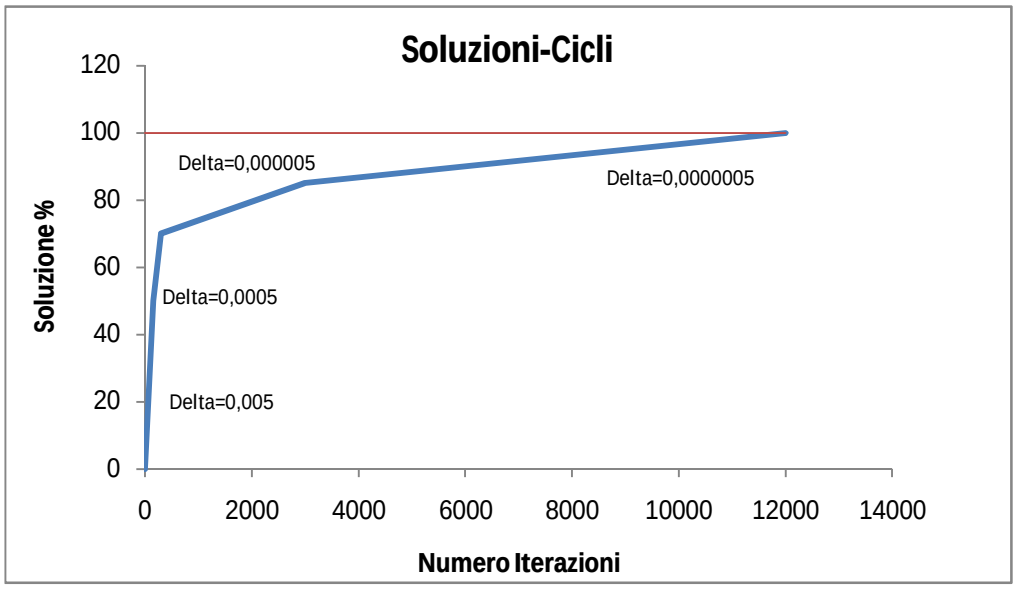


Figura 9, Iterazioni Convergenza-Efficacia DC

Nel grafico sull'asse delle ordinate si è pongono le soluzioni migliore trovate con ogni setting ed espresse come percentuale dell'ottimo globale definito sulle varie prove. Infatti con il setting più fine, si arriva alle soluzioni del 100%.

4. Maggiore è la complessità dei casi studiati (in particolare la dimensione globale delle linee e il buffer totale da allocare) e minore è la capacità dello strumento di fornire costantemente valori ottimali/sub ottimali. Se nei casi semplici il suo utilizzo porta sempre ai soliti risultati,anche nel 100% dei casi, con linee più elevate tale comportamento non viene più rispettato.

5. Dai casi studiati è emerso che utilizzando setting “grezzi”, l’algoritmo fornisce velocemente soluzioni di compromesso, mentre utilizzandolo in modalità “Fine Tuning” questo permetta di ottenere risultati migliori in un tempo più elevato. Per eseguire un’analisi completa di un caso potrebbe essere utile svolgere delle prove veloci con cui attestare il livello medio dei TP della linea per poi raffinare la ricerca in un secondo momento, con la sicurezza di partire da una buona base di ricerca.

3.3 Harmony Search

L' Harmony Search è un algoritmo di ottimizzazione di tipo metaeuristico che emula i comportamenti tipici dei musicisti nell'atto di composizione di nuove melodie.

Dunque così come nell'ambito musicale si ricerca costantemente di creare armonie originali, valutate in seguito da un pubblico estimatore, nell'ottimizzazione si tenterà di creare soluzioni possibili di un problema che permettono di risolverlo nel migliore dei modi.

Se in ambito musicale i nuovi brani vengono creati facendo si che vari strumenti suonino le giuste note allo stesso tempo, nei problemi di tipo matematico sarà fondamentale che le singole variabili di una soluzione siano settate su opportuni valori contemporaneamente.

Infine, così come la qualità della musica di una band migliora con l'esperienza e con la pratica, la funzione obiettivo di un problema può essere ottimizzata passo dopo passo tramite l'esplorazione e la conoscenza dello spazio delle sue soluzioni.

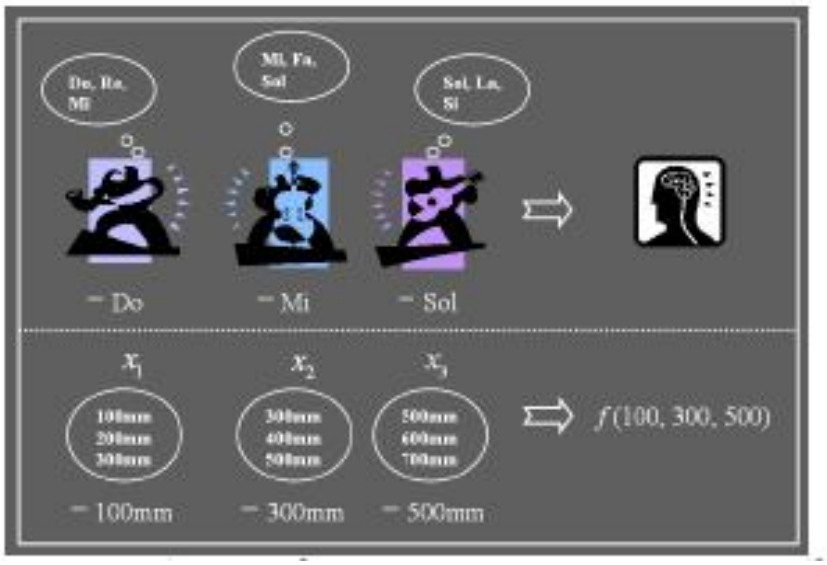


Figura 10,Analogie Generazione Melodie-Ottimizzazione

La figura 10 riassume l'analogia prima descritta, mettendo in evidenza in primis come ogni musicista rappresenti una variabile del problema da risolvere e in secondo luogo l'analogia fra il range di note che ogni strumento può suonare in determinato passaggio di un brano e i valori che una determinata variabile può effettivamente assumere all'interno di una determinata soluzione.

Elenchiamo di seguito i passi fondamentali dell' HS, riassunti ulteriormente in Figura 11:

- Inizializzazione dei parametri.
- Inizializzazione della memoria armonie(soluzioni iniziali)
- Improvvisazione di nuove armonie(creazione nuove soluzioni)
- Aggiornamento memoria armonie (introduzioni delle soluzioni candidate ottimo)
- Controllo del soddisfacimento del pubblico (check su criteri di stop dell'algoritmo)

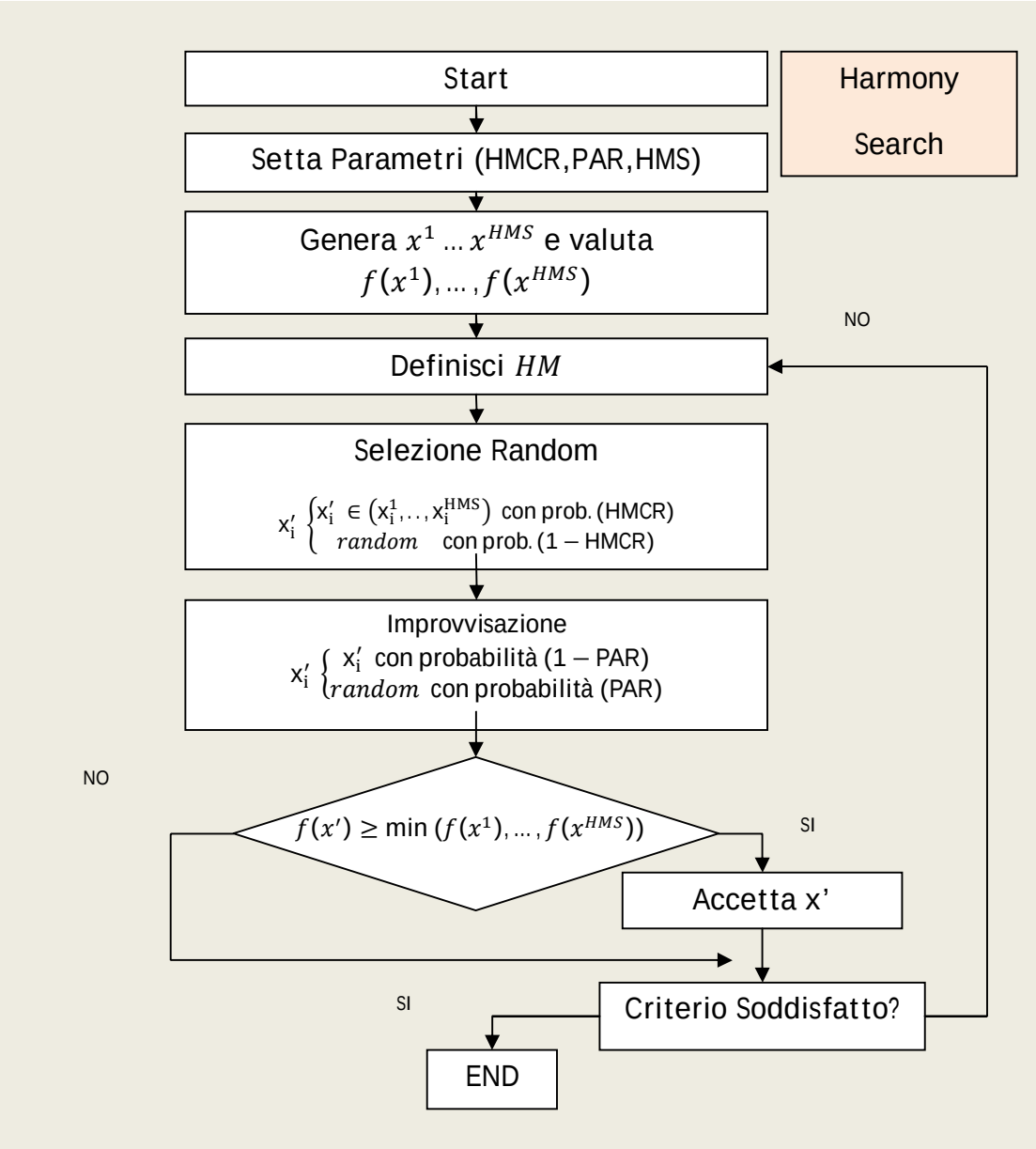


Figura 11, Flow Chart harmony Search

(i) **Inizializzazione dei parametri**

Questo è lo step in cui si formalizza il problema da risolvere, definendone:

- Funzione obiettivo $f(x)$ da min/massimizzare
- Struttura della soluzione: $x: \{x_{1,} \dots x_n\}\ con\ x_i \epsilon X_i$
- Vincoli del problema.
- HMS: Harmony Memory Size, grandezza del vettore che contiene le soluzioni possibili del problema
- HMCR: Harmony Memory Considerating Rate, parametro che identifica la probabilità di scegliere una nota fra le soluzioni comprese in $N(s)$.
- PAR: Pitch Adjusting Rate, parametro che regolamenta la fase di improvvisazione, identificando la probabilità con cui una nota sarà soggetta a variazione o meno in seguito alla fase della creazione di melodie base.
- NI:numero di improvvisazioni, ovvero numero di valutazioni dell'algoritmo

L'inizializzazione è un momento critico per il successo computazionale dell'algoritmo, perché ogni problema da affrontare necessiterà di appositi setting per essere risolto al meglio. La giusta identificazione dei valori dei parametri precedentemente illustrati può rendere l'algoritmo estremamente più efficace.
In prima approssimazione potremo dire che sebbene l'HS permetta per sua costituzione di esplorare a 360° lo spazio di ricerca delle soluzioni di un problema, è sempre di fondamentale importanza sostenere la ricerca stessa incanalando l'algoritmo sulla giusta strada.
Ad esempio una memoria $N(s)$ di dimensioni elevate potrebbe corrispondere alla possibilità di ricerca più elevata, ma meno efficace, o ancora impostare un PAR troppo elevato potrebbe voler dire difficoltà a perseguire velocemente una giusta direzione, sebbene in questo modo si dia la possibilità di esplorare una gamma di soluzioni più estesa.

(ii) Inizializzazione della memoria soluzioni

Il secondo passo consiste nel popolare la memoria soluzioni con un numero di referenza al più uguale all'HMS.

(iii)Improvvisazione di nuove armonie

A questo punto si crea una nuova soluzione x': $\{x'_{1,} \dots x_{,n}\}$ $con\ x'_i \epsilon X_i$ seguendo tre regole fondamentali:

- **Selezione Random**: così come i musicisti spesso ripropongono melodie ripescate dal loro repertorio, nel nostro caso sarà possibile riproporre note presenti in $N(s)$, pescate in maniera casuale all'interno delle possibili alternative. Il valore delle singole x_i' sarà scelto all'interno del loro dominio di appartenenza e logicamente con opportune regole di selezione.

$$x_i' \begin{cases} x_i' \in (x_i^1, .., x_i^{HMS}) \text{ con probabilità HMCR} \\ x_i' \ con\ probabilità\ (1 - HMCR) \end{cases}$$

- **Aggiustamento delle note:**una volta che tutte le note sono state selezionate, può accadere che il musicista decida di effettuare alcuni cambiamenti, per rendere la melodia ancora più originale, per adattarla al contesto, per rispettare determinati canoni di composizione. Allo stesso modo può accadere che una volta definito il vettore x', alcune note dello stesso vengano cambiate:

$$x_i' \begin{cases} x_i' = random \text{ con probabilità PAR} \\ x_i' \text{ con probabilità } (1 - PAR) \end{cases}$$

Il sistema di aggiustamento deve essere realizzato in modo tale che le note cambiate rispettino i vincoli imposti dalla melodia e questo può essere realizzato in due modi differenti:

- Si utilizza un sistema di improvvisazione intrinsecamente corretto (dunque che permetta immediatamente di scegliere una nota valida).
- Si effettua un'analisi a posteriori della correttezza della melodia creata, con cui le note che assumono valori non validi vengono rimodificate nel rispetto di tali vincoli.

Una terza opzione utilizzata, ma molto meno controllabile è quella di permettere alle variabili in gioco di poter assumere valori non validi, penalizzando poi la funzione obiettivo delle soluzioni per mezzo di opportuni fattori correttivi. Tale soluzione per quanto più facile da realizzare è sconsigliata perché l'algoritmo potrebbe convergere su soluzioni non valide che anche se penalizzate si trovano ad avere funzioni obiettivo migliori rispetto alle soluzioni valide.

(iv) Aggiornamento della memoria

Se la nuova soluzione creata in fase di composizione presenta una fitness function migliore rispetto a quelle delle soluzioni appartenenti a HM, ovvero migliora il risultato globale della ricerca, rientra automaticamente far parte della Harmony Memory, logicamente prendendo il posto della soluzione peggiore che si trova già al suo interno.

(v)Terminazione dell'algoritmo

L'algoritmo termina quando viene soddisfatta una determinata condizione, che solitamente corrisponde a :

$$N° \, Cicli > N^*$$

La condizione dunque impone un numero di cicli massimo all'HS per convergere oltre al quale si stima che l'algoritmo divenga inefficace.
Questo non esclude che a seconda dei casi la condizione di stop possa essere modificata in modo tale da rendere il più efficace ed efficiente l'algoritmo.

3.3.1 Harmony Search applicato al BAP

Anche nel caso dell'HS è stato necessario modificare l'algoritmo base per permettere che potesse essere applicato in maniera efficace al BAT. Di seguito sono descritti i cambiamenti realizzati.

Inizializzazione della HM

L'inizializzazione dell'HM avviene per mezzo di un algoritmo sviluppato appositamente che permette di creare soluzioni ammissibili del problema di tipo random.

Improvvisazione

La fase di improvvisazione è speculare a quella identificata nell'algoritmo classico eccetto per il fatto che ogni nota selezionata non potrà assumere valori qualsiasi, ma solo quelli compresi nell'intervallo $[0, B_{tot}]$.

Aggiustamento: questa fase non è contemplata nell'algoritmo di base, ma è stato necessario aggiungerla in quanto nel BAP esiste un vincolo di spazio massimo allocabile che deve essere sempre rispettato. Per farlo al termine di ogni ciclo di

creazione delle nuove soluzioni si valuta il valore di "spazio" attualmente allocato, sulle "note" che compongono x', che viene definito $Tetto$. A questo punto se accade che $Tetto$=B_{tot}, si ha una soluzione valida che entra direttamente nel ciclo di valutazione, se invece $Tetto \leq B_{tot}$, si alloca lo spread rimanente in modalità random sulle differenti posizioni di x_i', mentre se $Tetto \geq B_{tot}$ si elimina lo spread esistente in modalità random riducendo le componenti della soluzione non nulle.

Terminazione

L'algoritmo termina dopo un numero prestabilito di iterazioni.

3.3.2 Validazione del BHS (Buffer Harmony Search)

Una volta descritto a livello teorico il Buffer Harmony Search si è passati alla sua implementazione e alla successiva fase di test.

La validazione è stata realizzata per mezzo di un software applicativo, appositamente sviluppato in linguaggio Delphi: tale strumento sfrutta il modulo di valutazione di linee asincrone precedentemente realizzato per la validazione del MADDX.

L'obiettivo che si è voluto raggiungere è stato quello di comprendere come variano le prestazioni dell'HS al variare dei parametri fondamentali di computazione ovvero $HMS, HMCR, PAR\ e\ NC$.

Questo ha permesso inoltre di definire il setting ottimale di tali parametri con cui ottenere le migliori soluzioni durante la ricerca.

L'HS è uno strumento di ottimizzazione a tre parametri, ovvero l'HMR,il PAR e HMS.

Questo fatto implica una flessibilità molto elevata, che permette il suo utilizzo per una gamma di problemi estremamente elevata.

Tutto ciò si traduce però anche in una difficoltà intrinseca più elevata nella definizione del giusto settaggio, che gli permetta effettivamente di funzionare nel modo più efficace possibile.

E' stato deciso di utilizzare gli stessi esempi precedentemente testati per la validazione del DC, per comprendere come le prestazioni del BHS possano variare utilizzando valori dei parametri caratteristici differenti.

Logicamente per ogni esempio trattato sono state condotte gamme il più ampio possibile di prove, sia per mettere in risalto come effettivamente i differenti

parametri influiscono sul modello, sia per comprendere più a fondo il funzionamento dell'HS stesso.
Questa fase di test, per quanto lunga e basata su politiche del tipo "Trial ad Error", è risultata necessaria in quanto solamente riuscendo a garantire le migliori condizioni di funzionamento dell'HS si potrà in seguito realizzare una corretta analisi di benchmarking con il "Degraded Ceiling".
Il settaggio dell'algoritmo rappresenta la "scuola" di formazione del "musicista" Harmony Search, infatti a seconda dei valori scelti per tali parametri potremo avere "artisti" concreti con poche idee vincenti ,ma che avranno un successo limitato nel tempo, artisti mediocri con idee poco rilevanti e dunque destinati a non sfondare mai, altri che ricercano in continuazione, con poco successo ma che colpiscono nel segno una volta trovata la strada giusta.

Analisi dei dati

L'Harmony Search, testato su un caso semplice come la linea 1 risponde in maniera estremamente efficace e efficiente: impostando l'analisi su 250 cicli si riesce sempre a trovare la composizione che permette di arrivare all'ottimo globale.
L'esempio non è dunque molto rilevante ma ci ha permesso di comprendere che in generale volendo utilizzare un setting dei parametri con cui si da meno peso alle soluzioni presenti all'interno dell' Harmony Memory, è necessario incrementare i cicli di analisi per arrivare ad una buona soluzione finale.
La conseguenza logica è che diminuendo il fattore HMCR (dunque la probabilità di selezionare una nota dal HMB) per ottenere buoni risultati devo ridurre la grandezza di HMB perché in questo modo ogni volta che selezionerò una nota da tale insieme, questa rappresenterà effettivamente una buona candidata all'ottimo o almeno una buona indicatrice della strada giusta per raggiungerlo.

Caso	HMS	HMCR	PAR	Cicli	Vmin	Vmedio	Vmax	Prove
Linea 1.1	3	0,9	0,3	250	5,287009	5,287009	5,287009	100
Linea 1.2	3	0,8	0,2	200	5,287009	5,287009	5,287009	100
Linea 1.3	3	0,8	0,2	100	5,286832	5,286929	5,287009	100
Linea 1.4	3	0,7	0,2	250	5,287009	5,286997	5,287009	100
Linea 1.5	4	0,7	0,2	250	5,286983	5,286993	5,287009	100
Linea 1.6	1	0,7	0,2	250	5,28699	5,286995	5,287009	100
Linea 1.7	1	0,9	0,3	250	5,28699	5,286969	5,287009	100

Infine si può osservare che riducendo eccessivamente le dimensioni dell'HM, l'algoritmo necessita di un numero di iterazioni più elevate per convergere e dare risultati dello stesso livello .

La tabella mostra come su linee di dimensioni limitate l'HS presenta una facilità computazionale elevatissima, che lo porta sempre a raggiungere i medesimi risultati.

E' da sottolineare il fatto che negli esperimenti condotti, il numero di iterazioni globale è sempre stato abbondantemente sovrastimato (anche perché su linee di dimensioni ridotte, l'HS converge molto velocemente e dunque i tempi globali di esecuzioni dell'algoritmo non sono elevatissimi), fatto che ha portato all'ottenimento di risultati univoci.

Caso	HMS	HMCR	PAR	Cicli	Vmin	Vmedio	Vmax	Prove
Linea 2.1	3	0,9	0,3	200	3,781985	3,781985	3,781985	100
Linea 2.2	3	0,9	0,3	150	3,781985	3,781985	3,781985	100
Linea 2.3	3	0,9	0,3	100	3,781955	3,781979	3,781985	100
Linea 2.4	3	0,8	0,2	100	3,781955	3,781982	3,781985	100
Linea 2.5	4	0,8	0,2	200	3,781985	3,781985	3,781985	100
Linea 2.6	4	0,7	0,1	200	3,781985	3,781985	3,781985	100
Linea 2.7	4	0,7	0,1	250	3,781985	3,781985	3,781985	100
Linea 3.1	4	0,9	0,3	150	3,641206	3,641206	3,641206	100
Linea 3.2	4	0,8	0,2	250	3,641206	3,641206	3,641206	100
Linea 3.3	3	0,7	0,1	150	3,641206	3,641206	3,641206	100

Linea 3.1b	3	0,9	0,3	150	3,680802	3,680802	3,680802	100
Linea 3.2b	3	0,8	0,2	300	3,680802	3,680802	3,680802	100
Linea 3.3b	3	0,7	0,1	250	3,680802	3,680802	3,680802	100
Linea 3.4b	3	0,7	0,1	150	3,680802	3,680802	3,680802	100

Nel momento in cui vengono proposte due linee di tipo M5 di maggiore complessità, l'algoritmo perde la sua costanza nella restituzione dell'output e dunque il setting con cui viene mandato in esecuzione torna ad essere un fattore critico di successo per l'ottenimento di buone soluzioni.
Nei casi testati è emersa la capacità dell'HS di fornire soluzioni massimali di spicco, ma la sua caratteristica negativa risiede nel fatto che spesso manca di costanza: l'unico modo per riuscire a migliorare tale caratteristica è quello di puntare su un numero di cicli di valutazione nettamente superiore a discapito però dell'efficienza globale.
In sintesi si può dire che l'HS fatica nella fase di ricerca "a spanne", ma diviene particolarmente efficace nella fase di "Fine Tuning".
Una considerazione particolare va fatta sul caso 4.7, in cui si vede come incrementando eccessivamente le dimensioni di HMB, anche assicurando un numero di iterazioni molto elevate, si ottengono risultati poco rilevanti.
Questo fatto non dovrebbe stupire in quanto avere una memoria di dimensioni molto elevate di per se significa percorrere simultaneamente e a piccoli passi un numero di vie elevato. Questo comporta la necessità di un tempo nettamente superiore per intraprendere la via corretta verso l'ottimo.
Inoltre nei casi studiati la probabilità di selezionare note dalla memoria è ridotta, mentre la probabilità di PAR è molto elevata: in poche parole è come se l'algoritmo non "consultasse mai lo stradario" e procedesse alla cieca all'interno dello spazio delle soluzioni.
In definitiva è fondamentale assicurare che l'algoritmo segua una strada corretta, facendo si che la dimensione di HMB sia adeguata al numero di iterazioni globali e che la probabilità di cogliere nella memori a sia elevata. In parole povere bisogna assicurarci non solo che l'HS costruisca una buona "cartina" dello spazio da perlustrare, ma che questa venga effettivamente consultata.

Caso	HMS	HMCR	PAR	Cicli	Vmin	Vmedio	Vmax	Prove
Linea 4.1	3	0,9	0,3	150	3,472891	3,577173	3,700202	100
Linea 4.2	3	0,9	0,3	300	3,492805	3,577273	3,700202	100
Linea 4.3	3	0,9	0,3	600	3,492805	3,582497	3,700202	100
Linea 4.4	3	0,9	0,6	400	3,492805	3,532804	3,700202	100
Linea 4.5	5	0,7	0,6	400	3,489723	3,556627	3,700202	100
Linea 4.6	7	0,7	0,6	400	3,492805	3,492805	3,492805	100

La riprova di quanto detto si è avuta testando la linea 5, infatti imponendo gli stessi valori di HMCR e di PAR, nel momento in cui HMB diviene troppo capiente si ottengono risultati meno rilevanti.

Caso	HMS	HMCR	PAR	Cicli	Vmin	Vmedio	Vmax	Prove
Linea 5.1	3	0,9	0,3	600	4,659618	4,659955	4,660965	100
Linea 5.2	5	0,9	0,3	600	4,659618	4,659929	4,660965	100
Linea 5.3	4	0,9	0,6	600	4,659618	4,66783	4,703503	100

Muovendosi su linee del tipo M10, i comportamenti che avevamo sottolineato precedentemente divengono ancora più marcati, mettendo ulteriormente in evidenza il fatto che per assicurare valori massimali di spicco è fondamentale assicurare che le note componenti le nuove soluzioni vengano selezionate da HMB, mantenendo sempre una buona capacità di esplorazione, ovvero fissando PAR medio/alti.

Caso	HMS	HMCR	PAR	Cicli	Vmin	Vmedio	Vmax	Prove
Linea 6.1	3	0,9	0,6	1000	5,077353	5,204348	5,23075	100
Linea 6.2	3	0,8	0,6	1000	5,135851	5,198648	5,229689	100
Linea 6.3	4	0,7	0,6	1000	5,059018	5,195589	5,226728	100
Linea 7.1	3	0,85	0,4	1000	1,980416	2,229813	2,591504	100
Linea 7.2	3	0,85	0,86	1000	2,153826	2,230133	2,605404	100
Linea 8.1	3	0,9	0,6	1000	2,872332	3,018681	3,114828	100
Linea 8.2	3	0,8	0,4	1000	2,850537	2,978407	3,101152	100
Linea 8.3	3	0,7	0,4	1000	2,877637	3,030622	3,098875	100

Tale comportamento è mantenuto completamente anche su casi di dimensioni ancora più elevate quali linee del tipo M15 .

Caso	HMS	HMCR	PAR	Cicli	Vmin	Vmedio	Vmax	Prove
Linea 9.1	3	0,7	0,4	1000	5,020374	5,03897	5,076813	100
Linea 9.2	3	0,8	0,6	1000	4,996673	5,02367	5,043069	100
Linea 9.3	3	0,9	0,6	1000	4,996736	5,040481	5,145085	100

3.3.3 Riassunto caratteristiche del Buffer Harmony Search

Dall'analisi condotta precedentemente sono state individuate le seguenti caratteristiche peculiari dell'Harmony Search, proposte in maniera esplicita in modo tale che sia sempre possibile avere un riferimento al momento della definizione dei setting da impostare per l'utilizzo di tale algoritmo.

1. Minore è il peso dato alla soluzioni correnti di $N(s)$ e maggiore è il numero di cicli necessari per permettere all'algoritmo di convergere.
2. Maggiore è il numero di cicli impostati e più elevata è la probabilità di ottenere costantemente soluzioni ottime o sub-ottime.
3. HMS deve essere impostato in modo tale che HM non abbia né dimensioni eccessivamente ridotte, che ridurrebbero eccessivamente le possibilità di ricerca nello spazio delle soluzioni, né dimensioni eccessive che renderebbero dispersiva l'attività di spostamento dell'HS.
4. L'HMCR e il PAR devono essere sempre calibrati in modo tale da rendere la ricerca il più stabile possibile, in funzione di HMS e della dimensione della soluzione che si sta ricercando: il principio generale da adottare è quello per cui utilizzando HMC elevati si debbano assicurare valori di PAR medio/elevati. Infatti cogliendo sempre dalle note della memoria sarà necessario dare inerzia alla ricerca fornendo all'algoritmo nuovi valori delle componenti selezionate.
5. Più è elevata la dimensione globale delle soluzioni da cercare e minore è la capacità di HS di fornire costantemente soluzioni buone del problema a parità di iterazioni totali con cui viene mandato in esecuzione.

3.3.4 Advanced Harmony Search

Dai casi studiati è emerso che una volta impostato un determinato valore di HMS che stabilisce le dimensioni di HM, l'HMCR e il PAR rappresentano le due leve fondamentali su cui agire per rendere il più elevate possibile le prestazioni dell'HS. In realtà analizzando in maniera più approfondita l'algoritmo è stato dimostrato come esista un quarto parametro "fantasma" che incide fortemente sulle prestazioni globali dell'HS rappresentato dal sistema di modifica delle note durante la fase di improvvisazione. Attraverso degli esperimenti condotti sulle linee precedentemente proposte è stato dimostrato che a parità di setting, utilizzando un sistema di improvvisazione random che permette ad una nota di essere modificata su un ampio spazio di possibili valori si ottengono in generale soluzioni medie migliori, ma si inficia molto sull'ottenimento di valori massimali di spicco. Al contrario accade che l'algoritmo in generale fornisca soluzioni medie poco valide, ma raramente restituisca singoli valori di elevata fattura.
Questo fenomeno è facilmente spiegabile in quanto la prima opzione favorisce la fase di ricerca iniziale, che porta sostanzialmente nei pressi dell'ottimo, mentre la seconda, fa si che l'algoritmo esplori una porzione di spazio più limitata, ma le rare volte che viene a trovarsi nei pressi dell'ottimo vi si stabilisce senza più lasciarlo. L'idea è quella di utilizzare setting di HMCR e PAR equilibrati, come mostrato nel capitolo precedente e implementare un sistema che permetta di improvvisare le note in maniera sempre meno marcata durante le differenti fasi di esecuzione dell'algoritmo

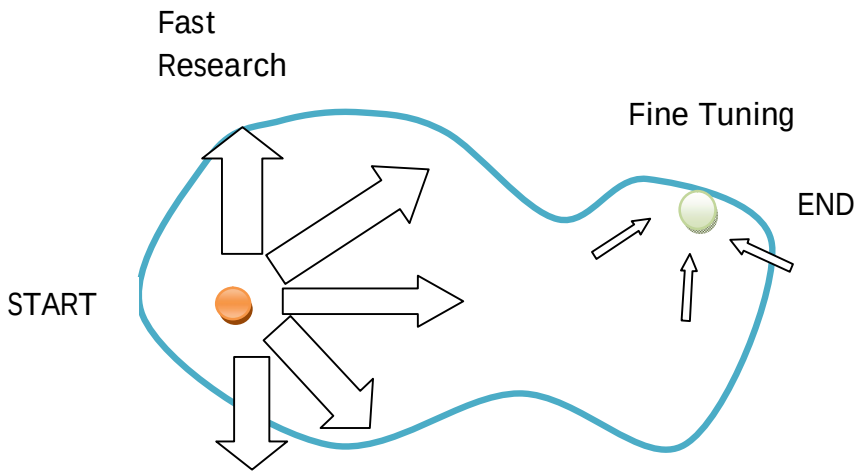


Figura 12, Punto di partenza Harmony sSearch

In questo modo potremo sfruttare efficacemente sia le fasi di Fast Research, che quelle di Fine Tuning.
La modifica proposta è stata implementata a livello software modificando il modulo di ottimizzazione relativo al'HS. I risultati ottenuti sono riassunti in tabella:

Caso	L1	L2	L3	L4	L5	L6	L7	L8	L9
HS	5,287	3,781	3,641	3,700	4,703	5,230	2,606	3,114	5,142
AHS	5,287	3,781	3,641	3,709	4,921	5,247	2,748	3,119	5,233

Il sistema ottenuto sarà utilizzato per tutte le successive fasi di ricerca.

3.4 Benchmarking

A questo punto della trattazione, avendo testato e validato le versioni del Degraded Ceiling e dell'Harmony Search relative al BAP, è possibile realizzare una analisi di Benchmarking per comprendere quale dei due algoritmi sia più indicato nella risoluzione di questo problema e che dunque permetta al nostro sistema di riuscire a fornire con più confidenza soluzioni ottime.
Come vedremo successivamente tale analisi si baserà su differenti fattori di prestazione, di cui il più importante sarà rappresentato dai valori massimi che tali algoritmi riusciranno a fornire per ogni caso studiato.
Il confronto fra i due algoritmi deve essere il più completo e vario possibile, perché potremmo riscontrare che non vi sia un "vincitore" univoco fra i due, ma che sotto determinate condizioni l'HS possa essere più efficace del DC e viceversa.
Avendo pianificato e documentato tutte le analisi precedentemente condotte per comprendere il funzionamento dei due sistemi è stato deciso di realizzare un'analisi principale sui dati già a nostra disposizione, che verranno integrati con un insieme di casi più particolari in un secondo momento.

3.4.1 Confronto Primario su casi test

Caso	V_{min} DC	V_{min} HS	Best	V_{med} DC	V_{med} HS	Best	V_{max} DC	V_{max} HS	Best
Linea1	5,287	5,287	Equiv.	5,287	5,287	Equiv.	5,287	5,287	Equiv.
Linea2	3,781	3,7819	Equiv.	3,7819	3,7819	Equiv.	3,7819	3,781	Equiv.
Linea 3.1	3,489	3,6412	HS	3,489	3,6412	HS	3,489	3,641	HS
Linea 3.2	3,526	3,6808	HS	3,527	3,6808	HS	3,526	3,680	HS
Linea 4	3,493	3,4928	DC	3,497	3,5827	HS	3,5	3,700	HS
Linea 5	4,66	4,6596	DC	4,677	4,6679	DC	4,703	4,703	HS
Linea 6	5,167	5,0773	DC	5,17	5,2043	HS	5,208	5,230	HS
Linea 7	2,453	2,1538	DC	2,429	2,2301	DC	2,6819	2,687	HS
Linea 8	0,991	2,8723	HS	1,661	3,0181	HS	3,056	3,114	HS
Linea 9	5,035	4,9967	DC	5,085	5,0401	DC	5,126	5,145	HS

Considerando le linee utilizzate per l'analisi e la sperimentazione condotta sul DC e sull'HS si arriva alle seguenti conclusioni:

- I due strumenti hanno prestazioni del tutto simili quando testati su casi semplici, sebbene l'HS arrivi più velocemente alla soluzione finale rispetto al DC.
- L'Harmony Search permette di arrivare a soluzioni globali migliori, dunque può essere momentaneamente il sistema più performante.
- Il Degraded Ceiling, sebbene presenti una capacità ridotta di fornire ottimi globali, grazie al suo sistema di aggiornamento di $N(s)$ e alla modalità di esplorazione a piccoli passi arriva più frequentemente a soluzioni medie "rilevanti",dunque può essere classificato più stabile.
- Per lo stesso motivo al punto precedente accade che l'HS spesso dia in output valori estremamente ridotti, sintomo di minore precisione.
- In tutti i casi testati l'HS richiede un tempo globale di valutazione inferiore rispetto al DC, dunque può essere ritenuto l'algoritmo più efficiente.

3.4.2 Confronto su ulteriori casi di studio

Per rendere la nostra analisi più esaustiva e precisa è stato deciso di sottoporre i due algoritmi a ulteriori prove facendo principalmente riferimento a casi complessi (che rappresentano al meglio la realtà), principalmente composti da linee costituite da un numero di macchine maggiore a dieci.

I valori dei parametri dei casi studiati sono stati posti nella sezione allegati.

Caso	B 1.1	B 1.2 210	B1 112	B2 112	B2 280	B2 210	B3 112	B3 140
Buffer	112	140	210	112	140	210	112	140
Vmax HS	5,691432	6,209002	6,064128	0,732612	1,263267	0,986517	1,326872	1,860066
Vmax DC	5,667144	6,155926	5,818658	0,706884	0,889471	0,704494	0,512605	0,782772

Caso	B3 210	B4 112	B4 210	B4 280	B5 190	B5 285	B5 380	B6 190
Buffer	210	112	140	210	190	285	380	190
Vmax HS	1,880033	1,450353	2,011456	2,212809	0,702092	0,882228	1,372099	1,372099
Vmax DC	1,627586	1,241121	1,487268	1,496973	0,686802	0,832297	1,049006	1,006512

Anche i seguenti esempi confermano quanto rilevato testando casi di complessità inferiore,mettendo in evidenza le prestazioni dei due algoritmi divergano all'aumentare della difficoltà dei casi proposti.

3.4.3 Conclusioni

A fronte dei dati ricavati dalle nostre analisi comparative, è possibile identificare l'HS come l'algoritmo di ottimizzazione più performante,da utilizzare per la risoluzione del problema di allocazione dei buffer su linee produttive asincrone. L'ottenimento nel 100% dei casi di soluzioni globali migliori rispetto a quelle ottenute con il DC fa si che possano essere non considerato completamente il fatto che il DC presenti una maggiore stabilità nel fornire soluzioni.

4 ToolBox

Il risultato finale a cui si è pervenuti per mezzo di questo elaborato è un Toolbox sviluppato partendo dai software pilota utilizzati durante le varie fasi di sviluppo e sperimentazione del nostro sistema di valutazione e ottimizzazione di linee produttive asincrone.

Tale strumento è stato sviluppato su piattaforma Borland, in linguaggio strutturato DELPHI e sintetizza tutte le funzioni illustrate nell'elaborato permettendo di:

- Valutare il Throughput globale di una linea di cui sono noti i parametri fondamentali e i valori delle capacità massime dei buffer compresi fra le macchine che la compongono.
- Ripartire un buffer di dimensioni B_{tot} fra i (k-1) buffer della linea per valutarne l'ottimo attraverso l'utilizzo dell'Harmony Search.
- Ripartire un buffer di dimensione B_{tot} fra i (k-1) buffer della linea per valutarne l'ottimo attraverso l'utilizzo del Degraded Ceiling.

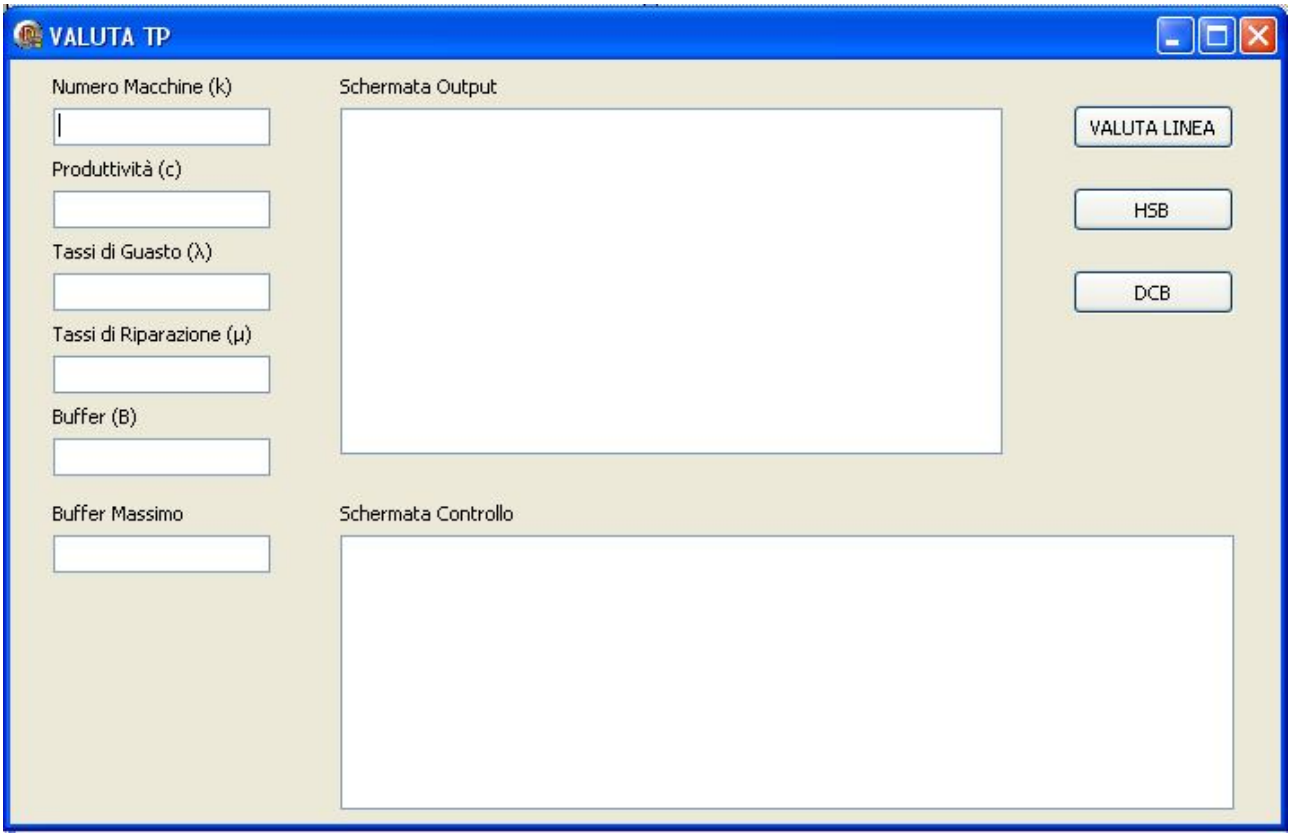


Figura 13, Schermata Principale Toolbox

Il Toolbox è piuttosto elementare, di facile comprensione e utilizzo, soprattutto grazie all'interfaccia grafica di cui è dotato.
Ogni volta che si voglia valutare/ottimizzare una linea è necessario impostare manualmente i parametri della stessa, definendo:

1. Il numero delle macchine che compongono la linea k
2. Impostare i tassi di guasto delle macchine inserendo nell'apposita casella un vettore in cui i tassi sono separati da spazi.
3. Impostare i tassi di riparazione inserendo nell'apposita casella un vettore in cui i dati sono separati da spazi e se decimali trascritti con il punto al posto della virgola.
4. Impostare le produttività inserendo nell'apposita casella un vettore in cui i dati sono separati da spazi e se decimali trascritti con il punto al posto della virgola.
5. Definire o lo spazio globale che si vuole suddividere fra i vari buffer della linea nel caso in cui se ne voglia ottimizzare le prestazioni, o una specifica suddivisione nel caso in cui si voglia semplicemente valutarne il Throughput in un caso specifico

4.1.1 Modulo di valutazione Throughput

Il valutatore restituisce, una volta impostati i dati sulla schermata centrale, il Throughput della linea descritta. Per avviarlo è necessario cliccare sull'apposito tasto nella schermata principale.
La chiamata di tale funzione fa si che sulla schermata di Controllo venga stampato l'iter evolutivo dell'algoritmo, che permette di comprendere come il MADDX sia arrivato alla soluzione finale.
La schermata di Output invece permette di visualizzare a schermo il Throughput calcolato per la linea stessa.

4.1.2 Modulo di ottimizzazione

Harmony Search

L'ottimizzatore basato sull'utilizzo dell'Harmony Search permette, una volta impostati i dati rappresentativi della linea, di valutare la composizione del buffer, che ottimizza il Throughput della linea.

Il suo utilizzo implica che sulla schermata iniziale sia impostato il valore del buffer totale che vogliamo allocare.
Il lancio avviene semplicemente tramite clic sull'apposito pulsante presente sulla schermata principale.

Output

L'ottimizzatore fornisce due output differenti: sulla schermata di controllo restituisce l'iter evolutivo che porta alle soluzioni trovate dall'ottimizzatore, mentre sulla schermata di Output questo restituisce le soluzioni trovate.
Ogni soluzione è composta da due informazioni differenti ovvero il valore del Throughput calcolato e il buffer a cui è effettivamente associato.

Degraded Ceiling

L'ottimizzatore basato sull'utilizzo del Degraded Ceiling permette, una volta impostati i dati rappresentativi della linea, di valutare la migliore composizione del buffer, che ottimizza il Throughput della linea.
Il suo utilizzo implica che sulla schermata iniziale sia impostato il valore del buffer totale che vogliamo allocare.
Il lancio avviene semplicemente tramite clic sull'apposito pulsante presente sulla schermata principale.

Output

L'ottimizzatore fornisce due output differenti: sulla schermata di controllo restituisce l'iter evolutivo che porta allee soluzioni trovate dall'ottimizzatore, mentre sulla schermata di Output questo restituisce le soluzioni trovate.
Ogni soluzione è composta da due informazioni differenti ovvero il valore del Throughput calcolato e il buffer a cui è effettivamente associato.

5 Conclusioni

5.1 Possibili utilizzi del Toolbox

Lo strumento da noi realizzato potrebbe essere impiegato come mezzo di supporto di processi di engineering e re-engineering di linee produttive: molte delle più grandi imprese mondiali investono quote ingenti dei loro fatturati (spesso anche più elevate del 5%) nelle attività di definizione dei layout per le loro linee produttive, necessari per realizzare nuovi prodotti.

Nel caso in cui una linea sia implementata ex novo il suo utilizzo permetterebbe di individuare immediatamente la migliore disposizione spaziale delle macchine nello stabilimento, in modo tale da assicurare ai buffer predisposti la giusta quantità di spazio con cui ottimizzare il sistema.

Nel caso di una linea da riprogettare il toolbox può permettere di comprendere effettivamente se una modifica dei buffer in essa presenti, può portare al miglioramento della resa del sistema e soprattutto comprendere se effettivamente tale modifica risulti essere economicamente vantaggiosa e dunque fattibile.

Esempio Pratico

Supponiamo di avere la seguente linea produttiva:

Macch.	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
λ	1/400	1/500	1/320	1/560	1/440	1/700	1/300	1/550	1/720	1/900	1/210	1/430	1/760	1/800	1/300
μ	1/20	1/30	1/12	1/23	1/24	1/25	1/26	1/16	1/28	1/21	1/7	1/8	1/40	1/33	1/12
c	8	9	9,2	9,3	8,5	8,2	9,3	9	8,1	8	9,3	8,7	8,4	8	9

Si dispone di uno spazio globale allocabile equivalente a 140 pezzi.

Attualmente la composizione dei buffer della linea è la seguente:

B_0	3 10 11 5 10 3 17 13 8 21 9 6 1 23

A cui corrisponde un Throughput uguale a 5,763736 $\left[\frac{Pezzi}{Minuto}\right]$

Tale valore è nettamente inferiore rispetto al riferimento asintotico della linea corrispondente alla produttività del collo di bottiglia uguale a 8 $\left[\frac{Pezzi}{Minuto}\right]$.

Decidiamo dunque di valutare se sia possibile ottenere dei miglioramenti ridefinendo il layout utilizzando il Toolbox prima descritto.

Dall'analisi si è rilevato che modificando la composizione delle capacità dei buffer è possibile portare il rendimento della linea a 6,064128 $\left[\frac{Pezzi}{Minuto}\right]$ con

B_1	25 19 23 11 4 17 0 1 1 10 3 14 10 2

Il costo delle attività di re-engineering si attesta sui 112.000 € ammortizzabili in due anni, mentre il prezzo di una singola unità venduta è di 0,76 €.
Non effettuando la modifica del layout il cash flow dell'impresa sarebbe di:

$$CF_0 = 5{,}76 * 60 * 8 * 7 * 52 * 2 = 1.437.696\ €$$

Realizzando la modifica:

$$CF_1 = 6{,}06 * 60 * 8 * 7 * 52 * 2 - (112.000) = 2.005.600\ €$$

La modifica in questione porterebbe ad un ricavo maggiore di

$$\Delta = (2.005.600 - 1.437.696) = 567.910\ €$$

Facendo logicamente ipotesi di vendere ogni pezzo prodotto in più.

5.2 Sviluppi Futuri

5.2.1 Utilizzo di Reti Neurali

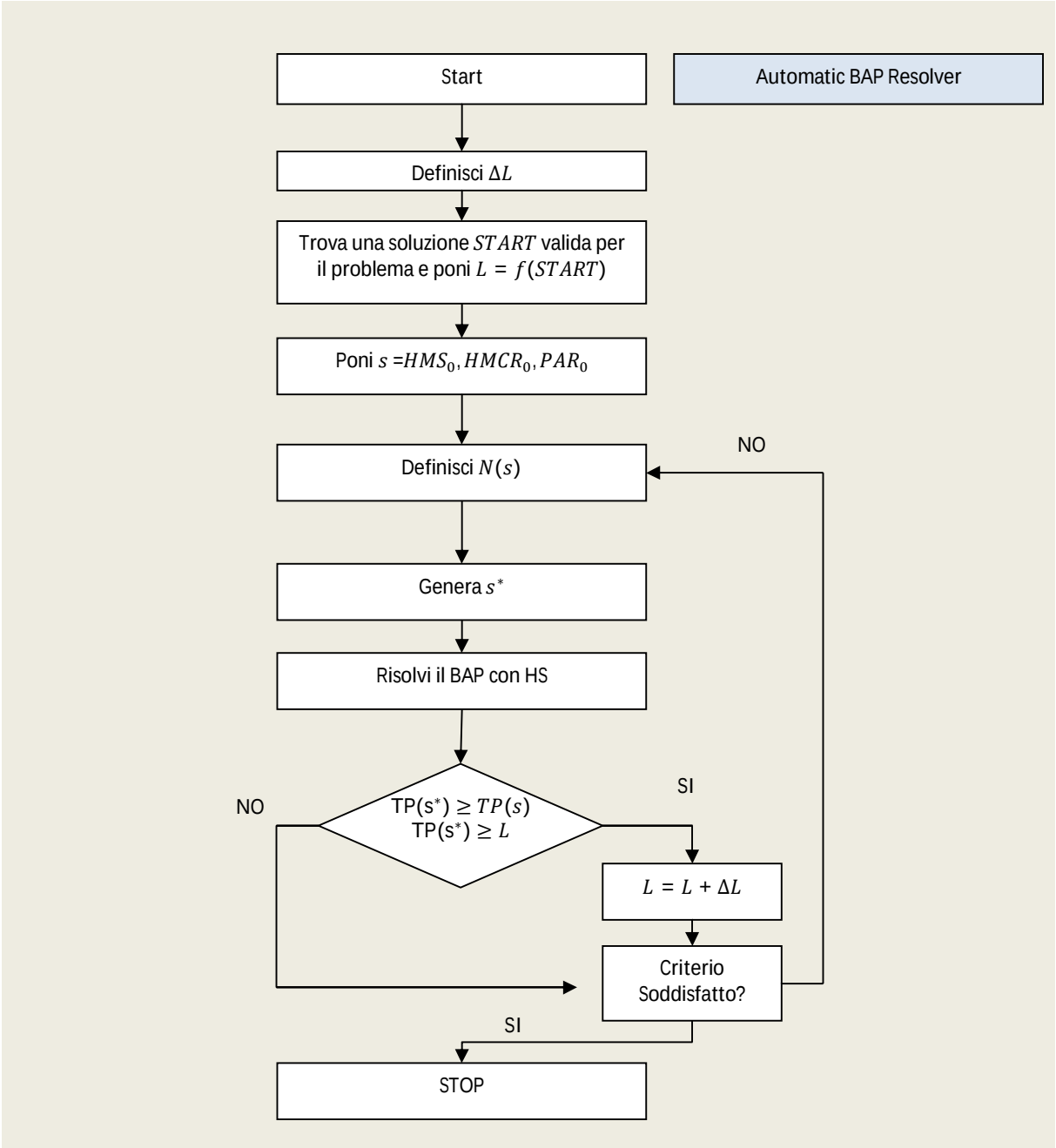
Nel capitolo iniziale della tesi è stato detto che ad oggi, gli unici due macrosistemi di valutazione del TP di una linea produttiva asincrona sono rappresentati dalla simulazione e dai sistemi analitici basati sviluppati partendo dalla teoria delle Catene di Markov.
Analizzando il comportamento tipico dell'Harmony Search nell'esplorazione delle possibili soluzioni del problema di allocazione dei buffer è nata l'idea di poter utilizzare un sistema alternativo di valutazione di linee produttive asincrone basato sull'utilizzo di Reti Neurali. L'analogia va ricercata nel criterio base di funzionamento dei due sistemi fondato sull'esperienza.
Le reti neurali sono sistemi software simili a delle black box, che in seguito ad opportune attività di training riescono a comprendere e riprodurre modelli matematici. L'allenamento di tali sistemi consiste nella somministrazione di "pacchetti" di analisi composti da Input del corrispondenti Output del sistema che si vuole replicare: dall'analisi di tali dati le reti neurali riescono ad apprendere il

funzionamento del modello e in seguito a riprodurlo, basandosi sulla loro esperienza.
Allenando dunque una rete con dati molto precisi derivanti dalla simulazione potremmo disporre in seguito di un sistema efficace e veloce.
Questo implica che una volta "imparato" il modello la rete neurale può essere utilizzata a sua volta per valutare nuovi dati di input e dunque nel nostro caso per poter valutare il TP di una linea.
Lo sviluppo di una rete neurale implicherebbe l'utilizzo nelle fase di training delle tecniche prima descritte per generare "buoni dati", con cui allenare la stessa, ma permetterebbe in seguito di utilizzare uno strumento di estrema rapidità e, nel caso in cui i dati fossero fittati correttamente, di elevata efficacia.
In definitiva potremmo ottenere uno strumento di valutazione "ideale" che permetta di unire la velocità dei sistemi di tipo analitico, con la precisione dei sistemi basati sulla simulazione.

5.2.2 Utilizzo del DC per settare al meglio l' HS

Il più grande problema raffrontato nella fase di ottimizzazione del TP delle linee testate è stato quello della ricerca, per ogni caso di studio, del miglior setting con cui mandare in esecuzione l'HS. Le difficoltà principali derivano dal fatto che tale attività è stata svolta manualmente, basandosi sull'intuito e portando avanti una politica di "Trial and Error". Il risultato finale è che tale attività è stata svolta in maniera inefficiente e molto probabilmente con risultati non ottimali avendo analizzato un numero molto limitato di casi.
Il matching fra i tre parametri dell'HS (senza contare le prove eseguite sui sistemi di aggiornamento del PAR o sul numero di iterazioni globali) ha richiesto una mole di tempo molto elevata, rallentando così le fasi di studio.
Un'idea molto ambiziosa sarebbe quella di riuscire a far svolgere questa fase in automatico agli stessi sistemi di ottimizzazione da noi usati per la risoluzione del BAT.
In particolare fissato l'HS come sistema ottimizzante per il BAT (in maniera concorde ai risultati ottenuti precedentemente) potremmo sfruttare il DC come ottimizzatore dei parametri dell'HS stesso. Infatti il DC risulta più facilmente settabile in quanto contraddistinto da un unico parametro (ΔL). Inoltre è stato dimostrato come tale strumento sia molto prestante quando debba definire

l'ottimo di sistemi di studio molto semplici (in questo caso i tre parametri dell'HS). In questo modo renderemmo il nostro strumento di analisi e ottimizzazione, non solo efficace, ma anche efficiente, limitando al minimo le attività manuali e assicurando con una confidenza ancora più elevata che i risultati ottenuti con il nostro sistema risultano essere effettivamente i migliori.



6 Riferimenti Bibliografici

- Gershwin S.B, Burman M.H, A decomposition Method for analyzing inhomogeneous assembly systems, Annals of Operations Research 93 (2000).
- Burman M.H., "New Results in Flow Line Analysis", PhD thesis MIT (2005)
- Nahas N., Ait-Kadi D., Nourelfalth M., " A new Approach fro buffer allocation in unreliable production lines", Production Economics 103 (2006)
- Dallery Y., David R., Xie X., "Approximate Analysis of Transfer Lines with Unreliable Machines and Finite Buffers", Transaction on automatic control. Vol.34 No.9 (1989).
- Burke E.K., Bykov Y., Newall J.P., "New Local Search Approach with Execution Time as an Input Parameter", Computer Science Technical Report.
- Li J., Meerkov S., "Production Systems Engineering", Springer (2009)
- Frosolini M.,Braglia M., Zammori F.A., " A Modified Harmony Search Algorithm for multi-ojective flowshop scheduling problem"
- Geem Z.W., Lee K.S., Park Y., "Application of Harmony Search to Vehicle Routing", American Journal of Applied Science 2 (2005).
- Geem Z.W., "Harmony Search Algorithm for Solving Sudoku", Springer (2007)
- Mahadvi M., Fesanghary M., Damangir E., " An Improved Harmony Search Algorithm for solving optimization problems", ScienceDirect Applied Mathematics anc Computation 188 (2007).

7 Appendici

7.1 Linee Test ADDX

Linea 1	**M1**	**M2**	**M3**	**M4**	**M5**
$\boldsymbol{\lambda}$	**0.1**	**0.2**	**0.1**	**0.2**	**0.3**
$\boldsymbol{\mu}$	**0.9**	**0.6**	**0.8**	**0.9**	**0.7**
$\boldsymbol{c}$	**5**	**5.3**	**4.9**	**5.2**	**4.7**

Linea 2	**M1**	**M2**	**M3**	**M4**	**M5**
$\boldsymbol{\lambda}$	**0.1**	**0.1**	**0.1**	**0.1**	**0.1**
$\boldsymbol{\mu}$	**0.9**	**0.9**	**0.9**	**0.9**	**0.9**
$\boldsymbol{c}$	**5**	**5.3**	**4.9**	**5.2**	**4.7**

Linea 3	**M1**	**M2**	**M3**	**M4**	**M5**
$\boldsymbol{\lambda}$	**0.1**	**0.1**	**0.1**	**0.1**	**0.1**
$\boldsymbol{\mu}$	**0.9**	**0.9**	**0.9**	**0.9**	**0.9**
$\boldsymbol{c}$	**1**	**1.3**	**0.8**	**0.9**	**1.2**

Linea 4	**M1**	**M2**	**M3**	**M4**	**M5**
$\boldsymbol{\lambda}$	**0.1**	**0.1**	**0.1**	**0.1**	**0.1**
$\boldsymbol{\mu}$	**0.9**	**0.9**	**0.9**	**0.9**	**0.9**
$\boldsymbol{c}$	**6.5**	**7**	**7.2**	**6.8**	**7.3**

7.2 Linee Test DC e HS

Linee M5

Linea 1	M1	M2	M3	M4	M5
$\boldsymbol{\lambda}$	(1/300)	(1/300)	(1/300)	(1/300)	(1/300)
$\boldsymbol{\mu}$	(1/10)	(1/10)	(1/10)	(1/10)	(1/10)
$\boldsymbol{c}$	6	6.3	6.2	6.1	6.2

Linea 2	M1	M2	M3	M4	M5
$\boldsymbol{\lambda}$	(1/400)	(1/300)	(1/500)	(1/400)	(1/300)
$\boldsymbol{\mu}$	(1/50)	(1/60)	(1/40)	(1/30)	(1/50)
$\boldsymbol{c}$	6	6.3	6.2	6.1	6.2

Linea 3	M1	M2	M3	M4	M5
$\boldsymbol{\lambda}$	(1/500)	(1/3000)	(1/300)	(1/4000)	(1/600)
$\boldsymbol{\mu}$	(1/70)	(1/80)	(1/100)	(1/120)	(1/80)
$\boldsymbol{c}$	7	6.3	8	6.5	7.3

Linea 4	M1	M2	M3	M4	M5
$\boldsymbol{\lambda}$	(1/500)	(1/300)	(1/300)	(1/400)	(1/600)
$\boldsymbol{\mu}$	(1/35)	(1/35)	(1/10)	(1/120)	(1/70)
$\boldsymbol{c}$	6	6.3	6.2	6.1	6.2

Linea 5	M1	M2	M3	M4	M5
$\boldsymbol{\lambda}$	(1/500)	(1/300)	(1/300)	(1/400)	(1/600)
$\boldsymbol{\mu}$	(1/35)	(1/35)	(1/10)	(1/120)	(1/70)
$\boldsymbol{c}$	9	8.3	8.2	9.1	8.5

Linee M10

Linea7	M1	M2	M3	M4	M5	M6	M7	M8	M9	M10
$\boldsymbol{\lambda}$	1/400	(1/300)	(1/500)	(1/400)	(1/300)	(1/200)	(1/300)	(1/300)	(1/400)	(1/200)
$\boldsymbol{\mu}$	(1/50)	(1/60)	(1/40)	(1/30)	(1/50)	(1/50)	(1/60)	(1/40)	(1/30)	(1/50)
$\boldsymbol{c}$	6	6.3	6.2	6.1	6.2	6	6.3	6.2	6.1	6.2

Linea6	M1	M2	M3	M4	M5	M6	M7	M8	M9	M10
$\boldsymbol{\lambda}$	(1/300	(1/300)	(1/300)	(1/300)	(1/300)	(1/300)	(1/300)	(1/300)	(1/300)	(1/300)
$\boldsymbol{\mu}$	(1/10)	(1/10)	(1/10)	(1/10)	(1/10)	(1/10)	(1/10)	(1/10)	(1/10)	(1/10)
$\boldsymbol{c}$	6	6.3	6.2	6.1	6.2	6.5	6.1	5.9	5.8	6.3

Linea8	M1	M2	M3	M4	M5	M6	M7	M8	M9	M10
$\boldsymbol{\lambda}$	1/500	(1/3000)	(1/300)	(1/4000)	(1/600)	(1/500)	(1/3000)	(1/300)	(1/4000)	(1/600)
$\boldsymbol{\mu}$	(1/70)	(1/80)	(1/100)	(1/120)	(1/80)	(1/70)	(1/80)	(1/100)	(1/120)	(1/80)
$\boldsymbol{c}$	7	6.3	8	6.5	7.3	7	6.3	8	6.5	7.3

Linea9	M1	M2	M3	M4	M5	M6	M7	M8	M9
$\boldsymbol{\lambda}$	(1/300	(1/300)	(1/300)	(1/300)	(1/300)	(1/300)	(1/300)	(1/300)	(1/300)
$\boldsymbol{\mu}$	(1/10)	(1/10)	(1/10)	(1/10)	(1/10)	(1/10)	(1/10)	(1/10)	(1/10)
$\boldsymbol{c}$	6	6.3	6.2	6.1	6.2	6.5	6.1	5.9	5.8

M10	M11	M12	M13	M14	M15
(1/300)	(1/300	(1/300)	(1/300)	(1/300)	(1/300)
(1/10)	(1/10)	(1/10)	(1/10)	(1/10)	(1/10)
6.3	6	6.3	6.2	6.1	6.2

7.3 Linee Benchmarking

B1	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
λ	1/400	1/500	1/320	1/560	1/440	1/700	1/300	1/550	1/720	1/900	1/210	1/430	1/760	1/800	1/300
μ	1/20	1/30	1/12	1/23	1/24	1/25	1/26	1\16	1/28	1/21	1/7	1/8	1/40	1/33	1/12
c	8	9	9,2	9,3	8,5	8,2	9,3	9	8,1	8	9,3	8,7	8,4	8	9

B2	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
λ	1/450	1/560	1/370	1/560	1/440	1/750	1/380	1/540	1/760	1/930	1/290	1/480	1/740	1/800	1/390
μ	1/60	1/60	1/52	1/43	1/54	1/75	1/26	1/46	1/58	1/61	1/37	1/28	1/65	1/55	1/43
c	12	12,5	11	10	14	13,2	11.09	15	13,6	14,2	13,9	14,2	12,7	13,6	14

B3	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
λ	1/450	1/560	1/370	1/560	1/440	1/750	1/380	1/540	1/760	1/930	1/290	1/480	1/740	1/800	1/390
μ	1/60	1/60	1/52	1/43	1/54	1/75	1/26	1/46	1/58	1/61	1/37	1/28	1/65	1/55	1/43
c	12	12,5	11	10	14	13,2	11.0	15	13,6	14,2	13,9	14,2	12,7	13,6	14

B4	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
λ	1/350	1/460	1/270	1/460	1/340	1/650	1/280	1/440	1/660	1/830	1/190	1/380	1/640	1/700	1/290
μ	1/70	1/70	1/62	1/53	1/64	1/85	1/36	1/46	1/68	1/71	1/47	1/38	1/65	1/55	1/43
c	12	12,5	11	10	14	13,2	11.09	15	13,6	14,2	13,9	14,2	12,7	13,6	14

B5	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
λ	1/350	1/460	1/270	1/460	1/340	1/650	1/280	1/440	1/660	1/830	1/190	1/380	1/640	1/700	1/290
μ	1/70	1/70	1/62	1/53	1/64	1/85	1/36	1/46	1/68	1/71	1/47	1/38	1/65	1/55	1/43
c	8	8,5	9	7	12	11,2	9,8	10	8,7	11	12	10,1	12	13,6	9.5

B6	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
λ	1/350	1/460	1/270	1/460	1/340	1/650	1/280	1/440	1/660	1/830	1/190	1/380	1/640	1/700	1/290
μ	1/70	1/70	1/62	1/53	1/64	1/85	1/36	1/46	1/68	1/71	1/47	1/38	1/65	1/55	1/43
c	12	12,5	11	10	14	13,2	11.09	15	13,6	14,2	13,9	14,2	12,7	13,6	14

16	17	18	19	20
1/190	1/270	1/500	1/430	1/6400
1/60	1/40	1/30	1/50	1/50
11	12,4	9,6	10,4	14

7.4 Codice Sorgente Toolbox

```
(*TOOLBOX OPERATIVO CHE RIASSUME LE FUNZIONI DI VALUTAZIONE DI LINEE
ASINCRONE E
  RISOLUZIONE DEL BUFFER ALLOCATION PROBLEM*)
unit toolbox;
interface
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls,
Forms,
  Dialogs, StdCtrls,Math;
type
  TForm2 = class(TForm)
    HSB: TButton;
    Mcontrollo: TMemo;
    DCB: TButton;
    val: TButton;
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    Label4: TLabel;
    Label5: TLabel;
    Edit1: TEdit;
    Edit2: TEdit;
    Edit3: TEdit;
    Edit4: TEdit;
    Edit5: TEdit;
    Label6: TLabel;
    Label7: TLabel;
    Mrisultati: TMemo;
    procedure HSBClick(Sender: TObject);
    procedure DCBClick(Sender: TObject);

  private
    { Private declarations }
  public
    { Public declarations }
    l,m,c:array of extended;
    N:array of integer;
    k:integer;

     scambio:real;
     scambioint:integer;

         end;

var
  Form2: TForm2;
   controllo:string;
s:string;
f:string;
w:string;
p:string;
z:integer;
i,j,k,B:integer;
LIMITE,provvisoria,delta,obiettivo:extended ;
NS:array of array of integer;
back1:boolean;
q:integer;
N:array of integer;
HMCR:extended;
sound:array of integer;
tetto:integer;
```

```
implementation

{$R *.dfm}

(*LA FUNZIONE VALUTA PREVEDE COME INPUT I VETTORI CARATTERISTICI DI UNA
LINEAN ASINCRONA
  DUNQUE IL NUMERO DI MACCHINE CHE LA COMPONGONO (k), I TASSI DI ROTTURA DI
OGNI MACCHINA (l),
  I TASSI DI RIPARAZIONE (m)  E LE PRODUTTIVITA' (c). INOLTRE VUOLE IN
INPUT LA COMPOSIZIONE
  DEI BUFFE (N) E RESTITUISCE IL VALORE DEL THROUGHPUT.*)

function valuta(k:integer;l,m,c:array of extended;N:array of
integer):extended;
var
(*nella versione iniziale mettiamo da tastiera,va messa subito qui perchè
poi k è usato per definire la dimensione di tutti i vettori*)
 (*battezziamo variabili di sistema,lambda,mu e produttività di ogni
macchina*)
buff:array of integer;(*capacità max di ogni buffer,valore corrente del
buffer*)
lu,mu,cu,ld,md,cd:array of real;(*battezziamo le variabili operative*)
E1,E2,E3,E4:real;
K1,K2:real;
F1,F2,F3,F4,F5,F6,F7,F8,F9: real;
G1,G2,G3,G4,G5,G6,G7,G8,G9:real;
eps,epsu,epsd:real;
fN10,fN01,PN11,PN10:real;
limite:real;
PR,C0,f010,f001,P001,P011:real;
TR: array of real;
A1,A2,B1,B2,O1,O2:real;
assoluto1:real;
assoluto2:real;
norma:array of real;
norma1,norma2:real;
risultato1:real;
risultato2:real;
x1,y1,z1,x2,y2,z2,x3,x4:real;
percent1:real;
percent2:real;
lctrl,cctrl,mctrl:real;
x,y:real;
j:integer;
contatore:integer;
varianza:real;
prova:integer;
lup,mup,cup,ldp,mdp,cdp,TRU,TRD:real;
back1,back2:boolean;
RISULTATI: extended;
q,r,t,g:integer;
somma:real;
varia:real;
soglia1,soglia2,soglia3,soglia4:real;
statistica1,statistica2:integer;



begin
   try
(*DESCRIZIONE DELLE MACCHINE*)
soglia4:=0.7;
```

```
statistica1:=0;
statistica2:=0;
q:=0;
setlength(TR,(k-1));
setlength(lu,(k-1));
setlength(mu,(k-1));
setlength(cu,(k-1));
setlength(ld,(k-1));
setlength(md,(k-1));
setlength(cd,(k-1));
setlength(norma,(k-1));
for z := 0 to k - 2 do
  begin
  TR[z]:=1;
  norma[z]:=1;
  end;
(*INIZIALIZZO LE VARIABILI PER SICUREZZA*)
E1:=1;
E2:=1;
E3:=1;
E4:=1;
K1:=1;
K2:=1;
F1:=1;
F2:=1;
F3:=1;
F4:=1;
F5:=1;
F6:=1;
F7:=1;
F8:=1;
F9:=1;
G1:=1;
G2:=1;
G3:=1;
G4:=1;
G5:=1;
G6:=1;
G7:=1;
G8:=1;
G9:=1;
eps:=1;
epsu:=1;
epsd:=1;
PR:=1;
C0:=1;
f010:=1;
f001:=1;
P001:=1;
P011:=1;
A1:=1;
A2:=1;
B1:=1;
B2:=1;
O1:=1;
O2:=1;
fN10:=1;
fN01:=1;
PN11:=1;
PN10:=1;
```

```
(*inizializziamo le variabili di lavoro di upstream*)
for z := 0 to  (k-2) do
begin (*inizializzo a spanne*)
lu[z]:=l[z];
mu[z]:=m[z];
cu[z]:=c[z];
end ;
(*ciclo per assegnare valori iniziali a ld*)
 for z := 0 to  (k-2) do
begin (*questo vuol dire che al 1° posto nella ld c'è il valore di
l(i+1)ecc.ecc..*)
ld[z]:=l[z+1];
md[z]:=m[z+1];
cd[z]:=c[z+1];
end ;
*ricontrollato--va bene*)
contatore:=0;
repeat
statistica1:=statistica1+1;
s:='';
f:='';
w:='';
p:='';
(*è corretto*)
for i:=0 to(k-3) do
begin
*inizio valutazione del TR secondo procedura di Gerwish*)
epsu:=(mu[i]/(mu[i]+lu[i]));
epsd:=(md[i]/(md[i]+ld[i]));
E1:=(ld[i]*(cu[i]*mu[i]+cd[i]*md[i])-md[i]*(cd[i]-
cu[i])*(lu[i]+mu[i]+ld[i]+md[i]))/(cu[i]*(cd[i]-cu[i])*(mu[i]+md[i]));

E2:=-(ld[i]*(cu[i]*mu[i]+cd[i]*md[i]))/(cu[i]*(cd[i]-cu[i])*(mu[i]+md[i]));

E3:=-(lu[i]*(cu[i]*mu[i]+cd[i]*md[i]))/(cd[i]*(cd[i]-cu[i])*(mu[i]+md[i]));

E4:=(lu[i]*(cu[i]*mu[i]+cd[i]*md[i])+mu[i]*(cd[i]-
cu[i])*(lu[i]+mu[i]+ld[i]+md[i]))/(cd[i]*(cd[i]-cu[i])*(mu[i]+md[i]));

K1:=((E1+E4)/2)+sqrt((sqr(E1-E4)+(4*E2*E3)))/2;

K2:=((E1+E4)/2)-sqrt((sqr(E1-E4)+(4*E2*E3)))/2;

if(cu[i]<cd[i]) then
F1:=(cd[i]*(mu[i]+md[i]+lu[i])*(E1-E4-
K1+K2)+(2*lu[i]*cu[i]*E2))/(cd[i]*(mu[i]+md[i]+lu[i])*(E1-E4+K1-
K2)+(2*lu[i]*cu[i]*E2))
else
F1:=(cd[i]*ld[i]*(E1-E4-
K1+K2)+(2*(mu[i]+md[i]+ld[i])*(cu[i]*E2)))/(cd[i]*ld[i]*(E1-E4+K1-
K2)+(2*(mu[i]+md[i]+ld[i])*(cu[i]*E2)));

F2:=E1-E4-(K1-K2);
F3:=E1-E4+(K1-K2);
F4:=(lu[i]/(mu[i]+md[i]))+(cd[i]/(cd[i]-cu[i]));
F5:=(ld[i]/(mu[i]+md[i]))-(cu[i]/(cd[i]-cu[i]));


if (cu[i]<cd[i])
then(*sintasssssiiii*)
```

```
F6:=((2*E2*cu[i])*(mu[i]+lu[i])*(mu[i]+md[i]+ld[i])*(1-(F1*exp((K1-
K2)*N[i])))+(cd[i]*ld[i])*(ld[i]+md[i])*(-F2+(F1*F3*(exp((K1-
K2)*N[i])))))/(2*E2*ld[i]*mu[i]*(lu[i]+mu[i]+ld[i]+md[i]))
else
F6:=(cd[i]*((F1*F3)-F2))/(2*mu[i]*E2);



if (cu[i]<cd[i])
then
F7:=(cu[i]*(1-F1)*(exp(K1*N[i])))/md[i]
else
F7:=(exp(K1*N[i]))*((cd[i]*(md[i]+ld[i])*(lu[i]+mu[i]+md[i])*(-
F2+(F1*F3*exp((K2-K1)*N[i])))))+(2*E2*cu[i]*lu[i]*(lu[i]+mu[i])*(1-
(F1*(exp((K2-K1)*N[i])))))/(2*E2*lu[i]*md[i]*(lu[i]+mu[i]+ld[i]+md[i]));

if (cu[i]<cd[i])
then
F8:=(((exp(K1*N[i]))-1)/K1)-((exp(K2*N[i])-1)*(exp((K1-K2)*N[i]))*F1)/K2
else
F8:=(((exp(K1*N[i]))-1)/(K1))-(((exp(K2*N[i])-1)*F1)/(K2));


if (cu[i]<cd[i])
then
F9:=(((1-exp(K1*N[i]))/K1)*(F2/(2*E2)))+(((exp(K2*N[i])-
1)/K2)*((F1*F3*exp((K2-K1)*N[i]))/(2*E2)))
else
F9:=(((1-exp(K1*N[i]))/K1)*(F2/(2*E2)))+(((exp(K2*N[i])-
1)/K2)*(F1*F3)/(2*E2));


G7:=sqrt(sqr((cu[i]*(mu[i]+md[i]+ld[i]))-
(cd[i]*(mu[i]+md[i]+lu[i])))+(4*cu[i]*cd[i]*lu[i]*ld[i]));
if(cu[i]<cd[i])then
G1:=(mu[i]*sqr(G7))+(mu[i]*G7)*((cu[i]*(mu[i]+md[i]+ld[i]))-
(cd[i]*(mu[i]+md[i]+lu[i])))
else
 G1:=(md[i]*sqr(G7))+(md[i]*G7)*((cu[i]*(mu[i]+md[i]+ld[i]))-
(cd[i]*(mu[i]+md[i]+lu[i])));


if (cu[i]<cd[i])
then
G2:=(md[i]*lu[i]*cd[i])*((cu[i]-cd[i])*(mu[i]-md[i])-
((cd[i]*lu[i])+(cu[i]*ld[i]))-G7)
else
G2:=(mu[i]*ld[i]*cu[i])*((cu[i]-cd[i])*(mu[i]-md[i])-
((cd[i]*lu[i])+(cu[i]*ld[i]))+G7);

if (cu[i]<cd[i])
then
G3:=(epsd*(cd[i]-(cu[i]*epsu))*G1+(cu[i]*epsu*(1-
epsd)*G2))/(cu[i]*epsu*(epsd-1))
else
G3:=(epsu*(cu[i]-(cd[i]*epsd))*G1+(cd[i]*epsd*(1-
epsu)*G2))/(cd[i]*epsd*(epsu-1));

if (cu[i]<cd[i])
then
G4:=cd[i]*epsd*G1
else
```

```
G4:=cu[i]*epsu*G1;

if (cu[i]<cd[i])
then
G5:=cu[i]*epsu*G2
else
G5:=cd[i]*epsd*G2;

if (cu[i]<cd[i])
then
G6:=cu[i]*epsu*G3
else
G6:=cd[i]*epsd*G3;

C0:=1/((F4*F8)+(F5*F9)+F6+F7);
f010:=C0*(1-(F1*exp((K1-K2)*N[i])));
f001:=C0*((-F2/(2*E2))+(F1*F3/(2*E2))*exp((K1-K2)*N[i]));

if(cu[i]<cd[i])
then
P011:=((cu[i]*(lu[i]+mu[i]+md[i])*f010)-
(cd[i]*ld[i]*f001))/(ld[i]*(lu[i]+mu[i]+ld[i]+md[i]))
else
P011:=0;

if(cu[i]<cd[i])
then
P001:=(mu[i]+md[i])*((cu[i]*lu[i]*f010)+(cd[i]*ld[i]*f001))/(ld[i]*mu[i]*(l
u[i]+mu[i]+ld[i]+md[i]))
else
P001:=((-cu[i]*f010)+(cd[i]*f001))/mu[i];
if(cu[i]<cd[i])then
TR[i]:=(G4+(G5*exp(-K2*N[i]))+(G6*exp(-K1*N[i])))/(G1+(G2*exp(-
K2*N[i]))+(G3*exp(-K1*N[i])))
else
TR[i]:=(G4+(G5*exp(K2*N[i]))+(G6*exp(K1*N[i])))/(G1+(G2*exp(K2*N[i]))+(G3*e
xp(K1*N[i])));

eps:=(m[i+1]/(m[i+1]+l[i+1]));
A1:=l[i+1]*((P011/TR[i])*((cu[i]/cd[i])-1))+mu[i]*(P001/TR[i]);
B1:=(mu[i]-m[i+1])*(P001/TR[i]);
O1:=1/((1/TR[i])+(1/(eps*c[i+1]))-(1/(epsd*cd[i])));

(*MODIFICA APPORTATA AL MADDX: HARD COSTRAINTS*)
(*Controllo N°1:lambdaup>0*)
lctrl:=((l[i+1]*B1*O1)+(m[i+1]*l[i+1])+(m[i+1]*A1*O1))/((m[i+1])+(B1*O1)-
(A1*O1));
if(lctrl>0)then
lu[i+1]:=lctrl
else
lu[i+1]:=l[i+1];

(*controllo N°2:mu>0*)
mctrl:= ((l[i+1]*B1*O1)+(m[i+1]*l[i+1])+(m[i+1]*A1*O1))/((l[i+1])+(A1*O1)-
(B1*O1));
 if(m[i+1]>m[i])then
begin
x:=m[i+1];
y:=m[i];
end
else
begin
```

```
y:=m[i+1];
x:=m[i];
end;
   if (mctrl>x)then
mu[i+1]:=x ;
if(mctrl<mu[i])then
mu[i+1]:=y;
if(mctrl>y)and (mctrl<x)then
mu[i+1]:=mctrl;



(*controllo N°3:produttività maggiore di zero e minore di m*)
cctrl:=(O1*(l[i+1]+m[i+1]))/((m[i+1])+(B1*O1)-(A1*O1));
if(cctrl>0)then
cu[i+1]:= cctrl
else
cu[i+1]:=cu[i+1];

s:=s+'  '+FLOATtoSTR(lu[i+1]);
f:=f+'  '+FLOATtoSTR(mu[i+1]);
w:=w+'  '+FLOATtoSTR(cu[i+1]);

end;

(*questo lavoro qui è fatto per lavorare solo con valori aggiornati*)
i:=k-2;
E1:=(ld[i]*(cu[i]*mu[i]+cd[i]*md[i])-md[i]*(cd[i]-
cu[i])*(lu[i]+mu[i]+ld[i]+md[i]))/(cu[i]*(cd[i]-cu[i])*(mu[i]+md[i]));

E2:=-(ld[i]*(cu[i]*mu[i]+cd[i]*md[i]))/(cu[i]*(cd[i]-cu[i])*(mu[i]+md[i]));

E3:=-(lu[i]*(cu[i]*mu[i]+cd[i]*md[i]))/(cd[i]*(cd[i]-cu[i])*(mu[i]+md[i]));

E4:=(lu[i]*(cu[i]*mu[i]+cd[i]*md[i])+mu[i]*(cd[i]-
cu[i])*(lu[i]+mu[i]+ld[i]+md[i]))/(cd[i]*(cd[i]-cu[i])*(mu[i]+md[i]));

K1:=((E1+E4)/2)+sqrt((sqr(E1-E4)+(4*E2*E3)))/2;

K2:=((E1+E4)/2)-sqrt((sqr(E1-E4)+(4*E2*E3)))/2;

if(cu[i]<cd[i])
then(*mmmmmmmm la sintassssiiii!*)
F1:=(cd[i]*(mu[i]+md[i]+lu[i])*(E1-E4-
K1+K2)+(2*lu[i]*cu[i]*E2))/(cd[i]*(mu[i]+md[i]+lu[i])*(E1-E4+K1-
K2)+(2*lu[i]*cu[i]*E2))
else
F1:=(cd[i]*ld[i]*(E1-E4-
K1+K2)+(2*(mu[i]+md[i]+ld[i])*(cu[i]*E2)))/(cd[i]*ld[i]*(E1-E4+K1-
K2)+(2*(mu[i]+md[i]+ld[i])*(cu[i]*E2)));

F2:=E1-E4-(K1-K2);
F3:=E1-E4+(K1-K2);
F4:=(lu[i]/(mu[i]+md[i]))+(cd[i]/(cd[i]-cu[i]));
F5:=(ld[i]/(mu[i]+md[i]))-(cu[i]/(cd[i]-cu[i]));

if (cu[i]<cd[i])
then(*sintasssssiiii*)
F6:=((2*E2*cu[i])*(mu[i]+lu[i])*(mu[i]+md[i]+ld[i])*(1-(F1*exp((K1-
K2)*N[i])))+(cd[i]*ld[i])*(ld[i]+md[i])*(-F2+(F1*F3*(exp((K1-
K2)*N[i])))))/(2*E2*ld[i]*mu[i]*(lu[i]+mu[i]+ld[i]+md[i]))
else
```

```
F6:=(cd[i]*((F1*F3)-F2))/(2*mu[i]*E2);

if (cu[i]<cd[i])
then
F7:=(cu[i]*(1-F1)*(exp(K1*N[i])))/md[i]
else
F7:=(exp(K1*N[i]))*((cd[i]*(md[i]+ld[i])*(lu[i]+mu[i]+md[i])*(-
F2+(F1*F3*exp((K2-K1)*N[i])))+(2*E2*cu[i]*lu[i]*(lu[i]+mu[i])*(1-
(F1*(exp((K2-K1)*N[i])))))/(2*E2*lu[i]*md[i]*(lu[i]+mu[i]+ld[i]+md[i]));

if (cu[i]<cd[i])
then
F8:=(((exp(K1*N[i]))-1)/K1)-((exp(K2*N[i])-1)*(exp((K1-K2)*N[i]))*F1)/K2
else
F8:=(((exp(K1*N[i]))-1)/(K1))-(((exp(K2*N[i])-1)*F1)/(K2));

if (cu[i]<cd[i])
then
F9:=(((1-exp(K1*N[i]))/K1)*(F2/(2*E2)))+(((exp(K2*N[i])-
1)/K2)*((F1*F3*exp((K2-K1)*N[i]))/(2*E2)))
else
F9:=(((1-exp(K1*N[i]))/K1)*(F2/(2*E2)))+(((exp(K2*N[i])-
1)/K2)*(F1*F3)/(2*E2));

G7:=sqrt(sqr((cu[i]*(mu[i]+md[i]+ld[i]))-
(cd[i]*(mu[i]+md[i]+lu[i])))+(4*cu[i]*cd[i]*lu[i]*ld[i]));
if(cu[i]<cd[i])then
G1:=(mu[i]*sqr(G7))+(mu[i]*G7)*((cu[i]*(mu[i]+md[i]+ld[i]))-
(cd[i]*(mu[i]+md[i]+lu[i])))
else
G1:=(md[i]*sqr(G7))+(md[i]*G7)*((cu[i]*(mu[i]+md[i]+ld[i]))-
(cd[i]*(mu[i]+md[i]+lu[i])));

if (cu[i]<cd[i])
then
G2:=(md[i]*lu[i]*cd[i])*((cu[i]-cd[i])*(mu[i]-md[i])-
((cd[i]*lu[i])+(cu[i]*ld[i]))-G7)
else
G2:=(mu[i]*ld[i]*cu[i])*((cu[i]-cd[i])*(mu[i]-md[i])-
((cd[i]*lu[i])+(cu[i]*ld[i]))+G7);

epsu:=(mu[i]/(mu[i]+lu[i]));
epsd:=(md[i]/(md[i]+ld[i]));

if (cu[i]<cd[i])
then
G3:=(epsd*(cd[i]-(cu[i]*epsu))*G1+(cu[i]*epsu*(1-
epsd)*G2))/(cu[i]*epsu*(epsd-1))
else
G3:=(epsu*(cu[i]-(cd[i]*epsd))*G1+(cd[i]*epsd*(1-
epsu)*G2))/(cd[i]*epsd*(epsu-1));

if (cu[i]<cd[i])
then
G4:=cd[i]*epsd*G1
else
G4:=cu[i]*epsu*G1;

if (cu[i]<cd[i])
then
G5:=cu[i]*epsu*G2
else
```

```
G5:=cd[i]*epsd*G2;

if (cu[i]<cd[i])
then
G6:=cu[i]*epsu*G3
else
G6:=cd[i]*epsd*G3;

C0:=1/((F4*F8)+(F5*F9)+F6+F7);
f010:=C0*(1-(F1*exp((K1-K2)*N[i])));
f001:=C0*((-F2/(2*E2))+(F1*F3/(2*E2))*exp((K1-K2)*N[i]));
fN10:=C0*((exp(K1*N[i]))-(F1*(exp((K1-K2)*N[i]))*(exp(K2*N[i]))));
fN01:=C0*((-F2/(2*E2))*(exp(K1*N[i]))+(F1*F3/(2*E2))*(exp((K1-
K2)*N[i]))*(exp(K2*N[i])));

if(cu[i]<cd[i])
then
PN11:=0
else
PN11:=((cd[i]*(lu[i]+mu[i]+md[i])*fN01)-
(cu[i]*lu[i]*fN10))/(lu[i]*(lu[i]+md[i]+ld[i]+mu[i]));

if(cu[i]<cd[i])
then
PN10:=((cu[i]*fN10)-(cd[i]*fN01))/md[i]
else
PN10:=(mu[i]+md[i])*((cu[i]*lu[i]*fN10)+(cd[i]*ld[i]*fN01))/(lu[i]*md[i]*(l
u[i]+mu[i]+ld[i]+md[i]));

if(cu[i]<cd[i])
then
P001:=(mu[i]+md[i])*((cu[i]*lu[i]*f010)+(cd[i]*ld[i]*f001))/(ld[i]*mu[i]*(l
u[i]+mu[i]+ld[i]+md[i]))
else
P001:=((-cu[i]*f010)+(cd[i]*f001))/mu[i];

if(cu[i]<cd[i])then
TR[i]:=(G4+(G5*exp(-K2*N[i]))+(G6*exp(-K1*N[i])))/(G1+(G2*exp(-
K2*N[i]))+(G3*exp(-K1*N[i])))
else
TR[i]:=(G4+(G5*exp(K2*N[i]))+(G6*exp(K1*N[i])))/(G1+(G2*exp(K2*N[i]))+(G3*e
xp(K1*N[i])));
s:='  ';
for i := 0 to k-2 do
begin
s:=s+'  '+FLOATtoSTR(TR[i]);
end;

(*qui finisce la valutazione di TR,adesso calcoliamo il parametro*)



(*a questo punto io ho tutti i valori di upstream*)
s:='';
f:='';
w:='';
(*il valore [0] lo trovo calcolando 1 ecc.ecc.*)

r:=0;
repeat
 for i:=1 to (k-2) do
begin
```

```
 (*inizio valutazione del TR secondo procedura di Gerwish*)
E1:=(ld[i]*(cu[i]*mu[i]+cd[i]*md[i])-md[i]*(cd[i]-
cu[i])*(lu[i]+mu[i]+ld[i]+md[i]))/(cu[i]*(cd[i]-cu[i])*(mu[i]+md[i]));

E2:=-(ld[i]*(cu[i]*mu[i]+cd[i]*md[i]))/(cu[i]*(cd[i]-cu[i])*(mu[i]+md[i]));

E3:=-(lu[i]*(cu[i]*mu[i]+cd[i]*md[i]))/(cd[i]*(cd[i]-cu[i])*(mu[i]+md[i]));

E4:=(lu[i]*(cu[i]*mu[i]+cd[i]*md[i])+mu[i]*(cd[i]-
cu[i])*(lu[i]+mu[i]+ld[i]+md[i]))/(cd[i]*(cd[i]-cu[i])*(mu[i]+md[i]));

K1:=((E1+E4)/2)+sqrt((sqr(E1-E4)+(4*E2*E3)))/2;

K2:=((E1+E4)/2)-sqrt((sqr(E1-E4)+(4*E2*E3)))/2;

if(cu[i]<cd[i])
then(*mmmmmmmm la sintassssiiii!*)
F1:=(cd[i]*(mu[i]+md[i]+lu[i])*(E1-E4-
K1+K2)+(2*lu[i]*cu[i]*E2))/(cd[i]*(mu[i]+md[i]+lu[i])*(E1-E4+K1-
K2)+(2*lu[i]*cu[i]*E2))
else
F1:=(cd[i]*ld[i]*(E1-E4-
K1+K2)+(2*(mu[i]+md[i]+ld[i])*(cu[i]*E2)))/(cd[i]*ld[i]*(E1-E4+K1-
K2)+(2*(mu[i]+md[i]+ld[i])*(cu[i]*E2)));



F2:=E1-E4-(K1-K2);
F3:=E1-E4+(K1-K2);
F4:=(lu[i]/(mu[i]+md[i]))+(cd[i]/(cd[i]-cu[i]));
F5:=(ld[i]/(mu[i]+md[i]))-(cu[i]/(cd[i]-cu[i]));


if (cu[i]<cd[i])
then(*sintasssssiiii*)
F6:=((2*E2*cu[i])*(mu[i]+lu[i])*(mu[i]+md[i]+ld[i])*(1-(F1*exp((K1-
K2)*N[i])))+(cd[i]*ld[i])*(ld[i]+md[i])*(-F2+(F1*F3*(exp((K1-
K2)*N[i])))))/(2*E2*ld[i]*mu[i]*(lu[i]+mu[i]+ld[i]+md[i]))
else
F6:=(cd[i]*((F1*F3)-F2))/(2*mu[i]*E2);



if (cu[i]<cd[i])
then
F7:=(cu[i]*(1-F1)*(exp(K1*N[i])))/md[i]
else
F7:=(exp(K1*N[i]))*((cd[i]*(md[i]+ld[i])*(lu[i]+mu[i]+md[i])*(-
F2+(F1*F3*exp((K2-K1)*N[i]))))+(2*E2*cu[i]*lu[i]*(lu[i]+mu[i])*(1-
(F1*(exp((K2-K1)*N[i])))))))/(2*E2*lu[i]*md[i]*(lu[i]+mu[i]+ld[i]+md[i]));

if (cu[i]<cd[i])
then
F8:=(((exp(K1*N[i]))-1)/K1)-((exp(K2*N[i])-1)*(exp((K1-K2)*N[i]))*F1)/K2
else
F8:=(((exp(K1*N[i]))-1)/(K1))-(((exp(K2*N[i])-1)*F1)/(K2));


if (cu[i]<cd[i])
then
```

```
F9:=(((1-exp(K1*N[i]))/K1)*(F2/(2*E2)))+(((exp(K2*N[i])-
1)/K2)*((F1*F3*exp((K2-K1)*N[i]))/(2*E2)))
else
F9:=(((1-exp(K1*N[i]))/K1)*(F2/(2*E2)))+(((exp(K2*N[i])-
1)/K2)*(F1*F3)/(2*E2));


G7:=sqrt(sqr((cu[i]*(mu[i]+md[i]+ld[i]))-
(cd[i]*(mu[i]+md[i]+lu[i])))+(4*cu[i]*cd[i]*lu[i]*ld[i]));

if(cu[i]<cd[i])then
G1:=(mu[i]*sqr(G7))+(mu[i]*G7)*((cu[i]*(mu[i]+md[i]+ld[i]))-
(cd[i]*(mu[i]+md[i]+lu[i])))
else
 G1:=(md[i]*sqr(G7))+(md[i]*G7)*((cu[i]*(mu[i]+md[i]+ld[i]))-
(cd[i]*(mu[i]+md[i]+lu[i])));


if (cu[i]<cd[i])
then
G2:=(md[i]*lu[i]*cd[i])*((cu[i]-cd[i])*(mu[i]-md[i])-
((cd[i]*lu[i])+(cu[i]*ld[i]))-G7)
else
G2:=(mu[i]*ld[i]*cu[i])*((cu[i]-cd[i])*(mu[i]-md[i])-
((cd[i]*lu[i])+(cu[i]*ld[i]))+G7);

epsu:=(mu[i]/(mu[i]+lu[i]));
epsd:=(md[i]/(md[i]+ld[i]));


if (cu[i]<cd[i])
then
G3:=(epsd*(cd[i]-(cu[i]*epsu))*G1+(cu[i]*epsu*(1-
epsd)*G2))/(cu[i]*epsu*(epsd-1))
else
G3:=(epsu*(cu[i]-(cd[i]*epsd))*G1+(cd[i]*epsd*(1-
epsu)*G2))/(cd[i]*epsd*(epsu-1));

if (cu[i]<cd[i])
then
G4:=cd[i]*epsd*G1
else
G4:=cu[i]*epsu*G1;


if (cu[i]<cd[i])
then
G5:=cu[i]*epsu*G2
else
G5:=cd[i]*epsd*G2;

if (cu[i]<cd[i])
then
G6:=cu[i]*epsu*G3
else
G6:=cd[i]*epsd*G3;

C0:=1/((F4*F8)+(F5*F9)+F6+F7);
f010:=C0*(1-(F1*exp((K1-K2)*N[i])));
f001:=C0*((-F2/(2*E2))+(F1*F3/(2*E2))*exp((K1-K2)*N[i]));
fN10:=C0*((exp(K1*N[i]))-(F1*(exp((K1-K2)*N[i]))*(exp(K2*N[i]))));
```

```
fN01:=C0*((-F2/(2*E2))*(exp(K1*N[i]))+(F1*F3/(2*E2))*(exp((K1-
K2)*N[i]))*(exp(K2*N[i])));

if(cu[i]<cd[i])
then
PN11:=0
else
PN11:=((cd[i]*(lu[i]+mu[i]+md[i])*fN01)-
(cu[i]*lu[i]*fN10))/(lu[i]*(lu[i]+md[i]+ld[i]+mu[i]));

if(cu[i]<cd[i])
then
PN10:=((cu[i]*fN10)-(cd[i]*fN01))/md[i]
else
PN10:=(mu[i]+md[i])*((cu[i]*lu[i]*fN10)+(cd[i]*ld[i]*fN01))/(lu[i]*md[i]*(l
u[i]+mu[i]+ld[i]+md[i]));

if(cu[i]<cd[i])
then
P001:=(mu[i]+md[i])*((cu[i]*lu[i]*f010)+(cd[i]*ld[i]*f001))/(ld[i]*mu[i]*(l
u[i]+mu[i]+ld[i]+md[i]))
else
P001:=((-cu[i]*f010)+(cd[i]*f001))/mu[i];

if(cu[i]<cd[i])then
TR[i]:=(G4+(G5*exp(-K2*N[i]))+(G6*exp(-K1*N[i])))/(G1+(G2*exp(-
K2*N[i]))+(G3*exp(-K1*N[i])))
else
TR[i]:=(G4+(G5*exp(K2*N[i]))+(G6*exp(K1*N[i])))/(G1+(G2*exp(K2*N[i]))+(G3*e
xp(K1*N[i])));
(*inizio a valutare i parametri di upstream*)
(*Attenzione: adesso ho calcolato il TR che è corretto, bisogna aggiornare
le variabili*)
(*Ripristiniamo le condizioni di partenza che serviranno per
l'aggiornamento*)

eps:=(m[i]/(m[i]+l[i]));
A2:=l[i]*((PN11/TR[i])*((cd[i]/cu[i])-1))+md[i]*(PN10/TR[i]);
B2:=(md[i]-m[i])*(PN10/TR[i]);
O2:=1/((1/TR[i])+(1/(eps*c[i]))-(1/(epsu*cu[i])));

(*ANCHE QUI IMPONGO GLI HARD COSTRAINTS*)
(*controllo N°1:lambdadown>0*)
lctrl:=((l[i]*B2*O2)+(m[i]*l[i])+(m[i]*A2*O2))/((m[i])+(B2*O2)-(A2*O2));
if(lctrl>0)then
ld[i-1]:=lctrl
else
ld[i-1]:=l[i];


(*controllo N°2:m>0*)
mctrl:=((l[i]*B2*O2)+(m[i]*l[i])+(m[i]*A2*O2))/((l[i])+(A2*O2)-(B2*O2));
 if(m[i]>md[i])then
 begin
   x:=m[i];
   y:=md[i];
 end
 else
 begin
   y:=m[i];
   x:=md[i];
 end;
```

```
 if(mctrl<y)then
 md[i-1]:=y;
 if(mctrl>x)then
 md[i-1]:=x;
 if(mctrl>y)and(mctrl<x)then
md[i-1]:=mctrl;



(*controllo N°3: cd>0*)
cctrl:= (O2*(l[i]+m[i]))/((m[i])+(B2*O2)-(A2*O2));
if(cctrl>0)then
cd[i-1]:=cctrl
else
cd[i-1]:= c[i];



s:=s+'  '+FLOATtoSTR(ld[i-1]);
f:=f+'  '+FLOATtoSTR(md[i-1]);
w:=w+'  '+FLOATtoSTR(cd[i-1]);
p:=p+'  '+FLOATtoSTR(TR[i]);
 end;
(*del ciclo for*);


  for i := 0 to k-2 do
    begin

(*QUESTA E' LA FASE CONCLUSIVA DEL CICLO DOVE CALCOLO TUTTI I TR AGGIORNATI
E LI CONFRONTO*)
 (*inizio valutazione del TR secondo procedura di Gerwish*)
E1:=(ld[i]*(cu[i]*mu[i]+cd[i]*md[i])-md[i]*(cd[i]-
cu[i])*(lu[i]+mu[i]+ld[i]+md[i]))/(cu[i]*(cd[i]-cu[i])*(mu[i]+md[i]));

E2:=-(ld[i]*(cu[i]*mu[i]+cd[i]*md[i]))/(cu[i]*(cd[i]-cu[i])*(mu[i]+md[i]));

E3:=-(lu[i]*(cu[i]*mu[i]+cd[i]*md[i]))/(cd[i]*(cd[i]-cu[i])*(mu[i]+md[i]));

E4:=(lu[i]*(cu[i]*mu[i]+cd[i]*md[i])+mu[i]*(cd[i]-
cu[i])*(lu[i]+mu[i]+ld[i]+md[i]))/(cd[i]*(cd[i]-cu[i])*(mu[i]+md[i]));

K1:=((E1+E4)/2)+sqrt((sqr(E1-E4)+(4*E2*E3)))/2;

K2:=((E1+E4)/2)-sqrt((sqr(E1-E4)+(4*E2*E3)))/2;
if(cu[i]<cd[i])
then(*mmmmmmmm la sintassssiiii!*)
F1:=(cd[i]*(mu[i]+md[i]+lu[i])*(E1-E4-
K1+K2)+(2*lu[i]*cu[i]*E2))/(cd[i]*(mu[i]+md[i]+lu[i])*(E1-E4+K1-
K2)+(2*lu[i]*cu[i]*E2))
else
F1:=(cd[i]*ld[i]*(E1-E4-
K1+K2)+(2*(mu[i]+md[i]+ld[i])*(cu[i]*E2)))/(cd[i]*ld[i]*(E1-E4+K1-
K2)+(2*(mu[i]+md[i]+ld[i])*(cu[i]*E2)));
F2:=E1-E4-(K1-K2);
F3:=E1-E4+(K1-K2);
F4:=(lu[i]/(mu[i]+md[i]))+(cd[i]/(cd[i]-cu[i]));
F5:=(ld[i]/(mu[i]+md[i]))-(cu[i]/(cd[i]-cu[i]));


if (cu[i]<cd[i])
then(*sintasssssiiii*)
```

```
F6:=((2*E2*cu[i])*(mu[i]+lu[i])*(mu[i]+md[i]+ld[i])*(1-(F1*exp((K1-
K2)*N[i])))+(cd[i]*ld[i])*(ld[i]+md[i])*(-F2+(F1*F3*(exp((K1-
K2)*N[i])))))/(2*E2*ld[i]*mu[i]*(lu[i]+mu[i]+ld[i]+md[i]))
else
F6:=(cd[i]*((F1*F3)-F2))/(2*mu[i]*E2);



if (cu[i]<cd[i])
then
F7:=(cu[i]*(1-F1)*(exp(K1*N[i])))/md[i]
else
F7:=(exp(K1*N[i]))*((cd[i]*(md[i]+ld[i])*(lu[i]+mu[i]+md[i])*(-
F2+(F1*F3*exp((K2-K1)*N[i])))))+(2*E2*cu[i]*lu[i]*(lu[i]+mu[i])*(1-
(F1*(exp((K2-K1)*N[i])))))))/(2*E2*lu[i]*md[i]*(lu[i]+mu[i]+ld[i]+md[i]));

if (cu[i]<cd[i])
then
F8:=(((exp(K1*N[i]))-1)/K1)-((exp(K2*N[i])-1)*(exp((K1-K2)*N[i]))*F1)/K2
else
F8:=(((exp(K1*N[i]))-1)/(K1))-(((exp(K2*N[i])-1)*F1)/(K2));


if (cu[i]<cd[i])
then
F9:=(((1-exp(K1*N[i]))/K1)*(F2/(2*E2)))+(((exp(K2*N[i])-
1)/K2)*((F1*F3*exp((K2-K1)*N[i]))/(2*E2)))
else
F9:=(((1-exp(K1*N[i]))/K1)*(F2/(2*E2)))+(((exp(K2*N[i])-
1)/K2)*(F1*F3)/(2*E2));


G7:=sqrt(sqr((cu[i]*(mu[i]+md[i]+ld[i]))-
(cd[i]*(mu[i]+md[i]+lu[i])))+(4*cu[i]*cd[i]*lu[i]*ld[i]));

if(cu[i]<cd[i])then
G1:=(mu[i]*sqr(G7))+(mu[i]*G7)*((cu[i]*(mu[i]+md[i]+ld[i]))-
(cd[i]*(mu[i]+md[i]+lu[i])))
else
G1:=(md[i]*sqr(G7))+(md[i]*G7)*((cu[i]*(mu[i]+md[i]+ld[i]))-
(cd[i]*(mu[i]+md[i]+lu[i])));


if (cu[i]<cd[i])
then
G2:=(md[i]*lu[i]*cd[i])*((cu[i]-cd[i])*(mu[i]-md[i])-
((cd[i]*lu[i])+(cu[i]*ld[i]))-G7)
else
G2:=(mu[i]*ld[i]*cu[i])*((cu[i]-cd[i])*(mu[i]-md[i])-
((cd[i]*lu[i])+(cu[i]*ld[i]))+G7);

epsu:=(mu[i]/(mu[i]+lu[i]));
epsd:=(md[i]/(md[i]+ld[i]));

if (cu[i]<cd[i])
then
G3:=(epsd*(cd[i]-(cu[i]*epsu))*G1+(cu[i]*epsu*(1-
epsd)*G2))/(cu[i]*epsu*(epsd-1))
else
G3:=(epsu*(cu[i]-(cd[i]*epsd))*G1+(cd[i]*epsd*(1-
epsu)*G2))/(cd[i]*epsd*(epsu-1));
```

```
if (cu[i]<cd[i])
then
G4:=cd[i]*epsd*G1
else
G4:=cu[i]*epsu*G1;


if (cu[i]<cd[i])
then
G5:=cu[i]*epsu*G2
else
G5:=cd[i]*epsd*G2;

if (cu[i]<cd[i])
then
G6:=cu[i]*epsu*G3
else
G6:=cd[i]*epsd*G3;

C0:=1/((F4*F8)+(F5*F9)+F6+F7);
f010:=C0*(1-(F1*exp((K1-K2)*N[i])));
f001:=C0*((-F2/(2*E2))+(F1*F3/(2*E2))*exp((K1-K2)*N[i]));
fN10:=C0*((exp(K1*N[i]))-(F1*(exp((K1-K2)*N[i]))*(exp(K2*N[i]))));
fN01:=C0*((-F2/(2*E2))*(exp(K1*N[i]))+(F1*F3/(2*E2))*(exp((K1-
K2)*N[i]))*(exp(K2*N[i])));

if(cu[i]<cd[i])
then
PN11:=0
else
PN11:=((cd[i]*(lu[i]+mu[i]+md[i])*fN01)-
(cu[i]*lu[i]*fN10))/(lu[i]*(lu[i]+md[i]+ld[i]+mu[i]));


if(cu[i]<cd[i])
then
PN10:=((cu[i]*fN10)-(cd[i]*fN01))/md[i]
else
PN10:=(mu[i]+md[i])*((cu[i]*lu[i]*fN10)+(cd[i]*ld[i]*fN01))/(lu[i]*md[i]*(l
u[i]+mu[i]+ld[i]+md[i]));


if(cu[i]<cd[i])
then
P001:=(mu[i]+md[i])*((cu[i]*lu[i]*f010)+(cd[i]*ld[i]*f001))/(ld[i]*mu[i]*(l
u[i]+mu[i]+ld[i]+md[i]))
else
P001:=((-cu[i]*f010)+(cd[i]*f001))/mu[i];


if(cu[i]<cd[i])then
TR[i]:=(G4+(G5*exp(-K2*N[i]))+(G6*exp(-K1*N[i])))/(G1+(G2*exp(-
K2*N[i]))+(G3*exp(-K1*N[i])))
 else

TR[i]:=(G4+(G5*exp(K2*N[i]))+(G6*exp(K1*N[i])))/(G1+(G2*exp(K2*N[i]))+(G3*e
xp(K1*N[i])));

    end;
 (*fine dell'aggiornamento,adesso si può passare alla fase di valutazione*)

 s:='  ';
```

```
for i := 0 to k-2 do
begin
s:=s+'  '+FLOATtoSTR(TR[i]);
end;
(*Qui finisce la valutazione di TR,adesso calcoliamo il parametro*)
(*PRIMA MODIFICA: DIFFERENTE CONDIZIONE DI STOP*)
somma:=0;
for i := 0 to k-2 do
somma:=somma+TR[i];
assoluto2:=(somma/(k-1));

for i:= 0 to k-2 do
norma[i]:= abs(assoluto2-TR[i]);
for i := 0 to k-3 do
begin
if(norma[i]>norma[i+1])then
norma[i+1]:=norma[i]
else
norma[i+1]:=norma[i+1];
end;
norma2:=norma[k-2];

contatore:=contatore+1;
r:=r+1;
(*ripetizione del ciclo di down*)
until(r>1)or(norma2<soglia4);

(*IMPONIAMO ANCHE UN SECONDO CRITERIO DI TIPO OPERATIVO CHE FA SI CHE
  NEL CASO IN CUI L'ALGORITMO NON DOVESSE CONCERGERE DI TERMINARE LA
COMPUTAZIONE
  DOPO UN NUMERO FINITO DI PASSI E ASSEGNARE UN VALORE NEGATIVO AL TR*)
until (contatore>70)or(norma2<soglia4);
s:='  ';
for i := 0 to k-2 do
s:=s+'  '+INTtoSTR(N[i]);


if(contatore>70)then
begin
RISULTATI:=-1

end;
if(norma2<soglia4)then
begin
RISULTATI:=assoluto2;

end;
Result:=RISULTATI;
except
on EZeroDivide do
Result:=-1;
on EOverflow do
Result:=-1
end;
end;


(*LA FUNZIONE SEGUENTE PERMETTE DI OTTIMIZZARE IL TP DI UNA LINEA PER MEZZO
DEL DEGRADED CEILING*)
procedure TForm2.DCBClick(Sender: TObject);
var
conto:integer;
```

```
j:integer;
B:integer;
media:extended;
finale:extended;
NF:array of integer;

begin
(*LUNGHEZZA DELLA LINEA*)
k:=20;
setlength(NF,k-1);
setlength(N,k-1);
setlength(l,k);
setlength(m,k);
setlength(c,k);
media:=3;
conto:=0;
s:='  ';
(*NEL CASO PILOTA FORNITO L'IMMISSIONE DEI PARAMETRI DELLE LINEE
  E' MANUALE*)
repeat
(*PROCESSO CON CUI SI INIZIALIZZA LE MACCHINE DELLA LINEA*)
for i := 0 to k-2 do
N[i]:=15;
(*creazione buffer in automatico*)
(*inizializzo le variabili*)
setlength(NS,1000,k-1);
delta:=0.00005;
provvisoria:=valuta(k,l,m,c,N); (*soluzione dell'ultima s*)
obiettivo:=valuta(k,l,m,c,N); (*tetto superiore*)
finale:=valuta(k,l,m,c,N);
for i:=0 to k-2 do
NS[0,i]:=N[i];
q:=0;
repeat
back1:=false;
repeat
(*CREAZIONE NUOVE SOLUZIONI RANDOM*)
i:=RandomRange(0,k-1);
j:=RandomRange(0,k-1);
if(i<>j)then
begin
if(N[i]>0)then
begin
N[i]:=N[i]-1;
N[j]:=N[j]+1;
back1:=true;
end;
end;
until(back1=true) ;  (*qui finisce l'aggiornamento*)

s:='  '; (*stringa vuota per i valori correnti del buffer*)

LIMITE:=valuta(k,l,m,c,N);(*valore di TR con l'allocazione di buffer
corrente*)
if LIMITE>=finale then
begin
finale:=LIMITE;
for i := 0 to k-2 do
NF[i]:=N[i];
end;

if(LIMITE>=provvisoria)or(LIMITE>=obiettivo)then
```

```
begin
for i := 0 to k-2 do
begin
NS[0,i]:=N[i];   (*la soluzione entra nel Neigh*)
s:=s+'  '+INTtoSTR(N[i]);(*mi stampo i valori del buffer*)
end;
obiettivo:=obiettivo+delta; (*incremento la funzione obiettivo*)
provvisoria:=LIMITE;   (*e dico che l'ultimo valore calcolato è questo*)
Mcontrollo.Lines.Add(FLOATtoSTR(provvisoria)+'  '+s);
//controllo per evidenziare a schemo cosa sta succedendo.
end;
*questo sarebbe l'else,ricomincio dall'ultima s valutata*)
for i := 0 to k-2 do
begin
N[i]:=NS[0,i];
end;
q:=q+1;
until(q>3000) ;
conto:=conto+1;
Mrisultati.Lines.Add(FLOATtoSTR(finale));
s:=' ';
for i := 0 to k-2 do
s:=s+'  '+INTtoSTR(NF[i]);
Mrisultati.Lines.Add(s);
until(conto>100)
(*COSI' GENERO 100 VALORI PER OGNI CAOS STUDIATO*)
end;


procedure TForm2.HSBClick(Sender: TObject);
var
HMB:array of array of integer;
HMF:array of extended;
hMS:integer;
differenza:integer;
attuale:extended;
conta:integer;
scelta:extended;
seleziona:integer;
i,j,z:integer;
PAR:extended;
contatore:integer;
begin
(*IMPOSTO NUMERO DI MACCHINE*)
k:=20;
setlength(N,k-1);
setlength(l,k);
setlength(m,k);
setlength(c,k);
Mcontrollo.Lines.Add('Lets start');
HMS:=4;
Mcontrollo.Lines.Add('HMS:  '+INTtoSTR(HMS));
(*NEL CODICE PILOTA FORNITO L'IMMISSIONE DEI PARAMETRI DELLE MACCHINE E'
MANUALE*)
l[0]:=(1/400);
l[1]:=(1/500);
l[2]:=(1/400);
l[3]:=(1/500);
l[4]:=(1/400);
l[5]:=(1/500);
l[6]:=(1/400);
l[7]:=(1/500);
```

```
l[8]:=(1/400);
l[9]:=(1/500);
l[10]:=(1/400);
l[11]:=(1/500);
l[12]:=(1/400);
l[13]:=(1/500);
l[14]:=(1/400);
l[15]:=(1/500);
l[16]:=(1/400);
l[17]:=(1/500);
l[18]:=(1/400);
l[19]:=(1/500);
(*l[20]:=(1/200);
l[21]:=(1/300);
l[22]:=(1/300);
l[23]:=(1/400);
l[24]:=(1/200);
l[25]:=(1/400);
l[26]:=(1/300);
l[27]:=(1/500);
l[28]:=(1/400);
l[29]:=(1/300); *)
m[0]:=(1/20);
m[1]:=(1/30);
m[2]:=(1/20);
m[3]:=(1/30);
m[4]:=(1/20);
m[5]:=(1/30);
m[6]:=(1/20);
m[7]:=(1/30);
m[8]:=(1/20);
m[9]:=(1/30);
m[10]:=(1/20);
m[11]:=(1/30);
m[12]:=(1/20);
m[13]:=(1/30);
m[14]:=(1/20);
m[15]:=(1/30);
m[16]:=(1/20);
m[17]:=(1/30);
m[18]:=(1/20);
m[19]:=(1/30);
(*m[20]:=(1/50);
m[21]:=(1/60);
m[22]:=(1/40);
m[23]:=(1/30);
m[24]:=(1/50);
m[25]:=(1/50);
m[26]:=(1/60);
m[27]:=(1/40);
m[28]:=(1/30);
m[29]:=(1/50); *)
c[0]:=(8);
c[1]:=(8.5);
c[2]:=(9);
c[3]:=(7);
c[4]:=(12);
c[5]:=(11.2);
c[6]:=(9.8);
c[7]:=10;
c[8]:=8.7;
c[9]:=11;
```

```
c[10]:=(12);
c[11]:=(10.1);
c[12]:=12;
c[13]:=13.6;
c[14]:=9.5;
c[15]:=(10);
c[16]:=(11);
c[17]:=(12.4);
c[18]:=(9.6);
c[19]:=(10.4);
(*c[20]:=(6);
c[21]:=(6.3);
c[22]:=6.2;
c[23]:=6.1;
c[24]:=6.2;
c[25]:=(6);
c[26]:=(6.3);
c[27]:=6.2;
c[28]:=6.1;
c[29]:=6.2; *)
(*setto le lunghezze delle variabili in gioco*)
 setlength(HMB,HMS);  //righe hanno la lunghezza del numero dei buffer

for i := 0 to HMS-1 do //colonne hanno la lunghezza dell'HMS
setlength(HMB[i],k-1);
setlength(HMF,HMS);
contatore:=0;
repeat
//inizializzo
for i := 0 to HMS - 1 do
(*CICLO DI GENERAZIONE RANDOM DELLE SOLUZIONI DI N(S)*)
begin
  B:=285;
    for j := 0 to k-2 do
    begin //i buffer sono lunghi k-1!
      if(i*(k-1))<=B then
      begin
        HMB[i,j]:=i;
        B:=B-i;
      end
      else
      begin
        HMB[i,j]:=0;
      end;
    end;

    for j := 0 to k-3 do
    begin
        HMB[i,j]:=HMB[i,j]+RandomRange(0,B+1);
        B:=B-HMB[i,j];

      end;
      HMB[i,k-2]:=HMB[i,k-2]+B;

  end;
   Mcontrollo.Lines.Add('Vaffanculo1');
//facciamo un piccolo controllino per sicurezza
  B:=285;
for i := 0 to HMS - 1 do
 begin
 s:='  ';
  begin
```

```
    for j := 0 to k-2 do
     s:=s+'  '+INTtoSTR(HMB[i,j]);

      Mcontrollo.Lines.Add(s);
  end;
  end;
 // stampa le soluzioni iniziali


(*TROVO LE CORRISPONDENTI FUNZIONI OBIETTIVO*)
 (*troviamo i corrispondenti valori di funzione*)
for j := 0 to HMS - 1 do
begin
for i := 0 to k-2 do
  N[i]:=HMB[j,i] ;
HMF[j]:=valuta(k,l,m,c,N);
end;
 //così abbiamo inizializzato anche la matrice delle funzioni

 //anche qui controllo tattico
 s:='  ';
 for i := 0 to HMS- 1 do
   s:=s+'  '+FLOATtoSTR(HMF[i]);
Mcontrollo.Lines.Add(s);

 //adesso dobbiamo ordinare le soluzioni
  repeat
  back1:=true;
  for i := 0 to (HMS - 2) do
  begin
    if(HMF[i]>HMF[i+1])then
    begin
      scambio:=HMF[i];
      HMF[i]:=HMF[i+1];
      HMF[i+1]:=scambio;
      for j := 0 to k-2 do
        begin
        scambioint:=HMB[i,j];
        HMB[i,j]:=HMB[(1+i),j];
         HMB[(1+i),j]:=scambioint;
        end;
    end;
  end;
    for i := 0 to HMS-2 do
    begin
      if(HMF[i]>HMF[i+1])then
      back1:=false;

    end;
  until(back1=true) ;
//funziona fino a qui
s:='  ';

 //controllo per vedere se gli ordina
 for i := 0 to HMS- 1 do
   s:=s+'  '+FLOATtoSTR(HMF[i]);
Mcontrollo.Lines.Add(s);
 for i := 0 to HMS - 1 do
 begin
 s:='  ';
  begin
    for j := 0 to k-2 do
```

```
     s:=s+'  '+INTtoSTR(HMB[i,j]);

      Mcontrollo.Lines.Add(s);
  end;
  end;
  //da ricontrollare
//inizia la fase di creazione della nuova soluzione
 //random:
 conta:=0;
 repeat
  HMCR:=0.85;
 setlength(sound,k-1);
 for i := 0 to (k-2) do
 begin
  scelta:=Random;

  if(scelta<=HMCR)then
  begin
  seleziona:=RandomRange(0,(HMS));//HMS perchè l'ultima postazione ha
posizione HMS-1
  Mcontrollo.Lines.Add('seleziona:=  '+FLOATtoSTR(seleziona));
  sound[i]:=HMB[seleziona,i];
  end
  else
  sound[i]:=-100;
 end;
 s:='  ';

 for i := 0 to k-2 do
 s:=s+'  '+INTtoSTR(sound[i]);
Mcontrollo.Lines.Add(s);
//ricontrollato anche qui funge


  //aggiustamento
 PAR:=0.3;
for i := 0 to k-2 do
  if sound[i]>=0 then
  begin
  scelta:=Random;
  Mcontrollo.Lines.Add('scelta:=  '+FLOATtoSTR(scelta));
  if(scelta>PAR)then
  sound[i]:=sound[i]+RandomRange(-20,4);
  end;
  s:='  ';
  for i := 0 to k-2 do
 s:=s+'  '+INTtoSTR(sound[i]);
Mcontrollo.Lines.Add(s);
 //anche questo funziona
 //adesso ho la nuova soluzione, ma è parziale. devo vedere se viola
vincoli
 //e dunque devo aggiustarla.
 tetto:=0;
 for i := 0 to k-2 do
   if(sound[i]>0)then
   tetto:=tetto+sound[i];
Mcontrollo.Lines.Add('Tetto:=  '+INTtoSTR(tetto));

//caso 1 :tetto uguale al Buffer Max
 if tetto=B then
 begin
  for i := 0 to k-2 do
```

```
   begin
     if(sound[i]<0)then
     sound[i]:=0;
   end;
 end;

 //caso2 alloco meno buffer rispetto al dovuto
if tetto<B then
begin
   for i := 0 to k-2 do
   begin
     if(sound[i]<0)then
     sound[i]:=0;
   end;

differenza:=B-tetto;
repeat
 i:=RandomRange(0,k-1);
 begin
 sound[i]:=sound[i]+1;
 differenza:=differenza-1;
 end;
until(differenza=0) ;
end;

//caso 3 alloco più buffer
 if tetto>B then
 begin
   for i := 0 to k-2 do
   begin
     if(sound[i]<0)then
     sound[i]:=0;
   end;

 differenza:=tetto-B;
repeat
 i:=RandomRange(0,k-1);
 if(sound[i]>0)then
 begin
 sound[i]:=sound[i]-1;
 differenza:=differenza-1
 end;
until(differenza=0) ;
 end;
 s:='  ';
 for i := 0 to k-2 do
 s:=s+'  '+INTtoSTR(sound[i]);
Mcontrollo.Lines.Add('sound'+'  '+s);
//fino a qui funziona

 //la soluzione c'è tutta,adesso la valutiamo
   attuale:=valuta(k,l,m,c,sound);
   Mcontrollo.Lines.Add('attuale=  '+FLOATtoSTR(attuale));
 //adesso aggiorniamo la matrice dei buffer:
 if(attuale>HMF[0])then
 begin
 back1:=true;
 for i := 0 to HMS-1 do
   begin
     if(attuale=HMF[i])then
     back1:=false;
   end;
```

```
 if(back1=true)then
  begin

 HMF[0]:=attuale;
 for i := 0 to k-2 do
   HMB[0,i]:=sound[i];
 end;
 end;
 //e la riordiniamo
  for i := 0 to (HMS - 2) do
    if(HMF[i]>HMF[i+1])then
    begin
      scambio:=HMF[i];
      HMF[i]:=HMF[i+1];
      HMF[i+1]:=scambio;
      for j := 0 to k-2 do
        begin
        scambioint:=HMB[i,j];
        HMB[i,j]:=HMB[(1+i),j];
         HMB[(1+i),j]:=scambioint;
        end;
  end;
 //aggiorno contatore
  conta:=conta+1;
  s:='  ';
   for i := 0 to HMS- 1 do
   s:=s+'  '+FLOATtoSTR(HMF[i]);
Mcontrollo.Lines.Add(s);
  for i := 0 to HMS - 1 do
 begin
 s:='  ';
  begin
    for j := 0 to k-2 do
     s:=s+'  '+INTtoSTR(HMB[i,j]);

      Mcontrollo.Lines.Add(s);
  end;
 end;

until(conta>950) ;
 Mrisultati.Lines.Add(FLOATtoSTR(HMF[HMS-1]));
 s:='  ';
 for i := 0 to k-2 do
    s:=s+'  '+INTtoSTR(HMB[(HMS-1),i]);
    Mrisultati.Lines.Add(s);
    contatore:=contatore+1;
              until(contatore>10) ;
end;

end.
```

Ringraziamenti

Questo lavoro segna la conclusione dei cinque anni più belli, intensi e difficili della mia vita, che non avrei mai potuto affrontare senza il sostegno di tutte le persone speciali che mi circondano.
Desidero innanzitutto ringraziare il Professor Braglia per i preziosi insegnamenti e per aver creduto sempre in me. Inoltre ringrazio sentitamente l'Ing. Frosolini e l'Ing. Zammori per la grandissima disponibilità e il tempo che mi hanno dedicato in questi anni.
Un ringraziamento speciale va ai miei genitori che mi sono stati vicino e mi hanno dato la forza di andare sempre avanti, vivendo in prima persona i "drammi" di un isterico studente fuori sede. Grazie per avermi insegnato ciò che non si può imparare su un banco di scuola e avermi permesso di vivere sempre nelle migliori condizioni.
Grazie ai miei angeli, Alessio ed Andrea, senza i quali non riuscirei a vivere: non ci sono parole per descrivere quanto importanti siate per me e quanto vi voglia bene.
Ringrazio di cuore la mia ragazza Ilaria, che da sempre mi sta accanto e mi supporta nei momenti peggiori. Ogni giorno mi regala il sorriso e riesce a comprendermi, rendendo la mia vita speciale: non esistono libri che insegnano la sincerità e l'amore.
Un grazie dovuto va a Cecilia e Lorenzo per tutti i momenti belli passati insieme e per aver dato un senso alla parola amicizia.
Non posso non ringraziare Serena, con cui condivido angosce scolastiche da dieci anni e che per me è sempre stata un punto di riferimento e un modello da emulare. La tua disponibilità e la tua amicizia non potranno mai essere ripagati.
In ultimo luogo vorrei esprimere la mia gratitudine ai miei amici e ai miei colleghi, in particolare a Marco per aver reso ancora più calda la mia casa e a Caterina per i bei momenti passati fra le mura di Ingegneria.